CARNEGIE MELLON

Department of Electrical and Computer Engineering

AFOSR-TR- 89-0440

Carnegie Mellon

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 89-0440
6a. NAME OF PERFORMING ORGANIZATION: Carnegie Mellon University; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: AFOSR/NE
6c. ADDRESS (City, State, and ZIP Code): Department of Electrical and Computer Engg. Center for Excellence in Optical Data Processing Pittsburgh, PA 15213	7b. ADDRESS (City, State, and ZIP Code): BLDG. 410 BAFB, DC 20332
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR/NE; 8b. OFFICE SYMBOL (If applicable): NE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-84-0293
8c. ADDRESS (City, State, and ZIP Code): Building 410 Bolling Air Force Base Washington, D.C. 20332-6448	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. 61102F; PROJECT NO. 2305; TASK NO. B1; WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Associative Processors and Directed Graphs for Optical Processing

12. PERSONAL AUTHOR(S)
David Casasent

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 8/84 TO 3/89	1989 February 15	145

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
			Associative processors, Computer generated holograms, Directed graphs, Distortion-invariant, Feature Extraction, Heteroassociative processors, Hierarchical processors, Ho-Kashyap (cont.)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This represents the final report on our recent 4 year effort. We briefly review our prior feature space and distortion-invariant processors. We then highlight several of our optical ~~AI~~ processor concepts. Emphasis is given to new associative processor and directed graph optical systems for large knowledge base processing.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: ~~David Casasent, Principal Investigator~~ GILES	22b. TELEPHONE (Include Area Code): ~~412-268-2464~~ (202) 767-4931; 22c. OFFICE SYMBOL: NE

ASSOCIATIVE PROCESSORS AND DIRECTED GRAPHS FOR OPTICAL PROCESSING

Final Report

Submitted to:
AFOSR/NE
Building 410
Bolling Air Force Base
Washington, D.C. 20332-6448
Attention: Dr. C. Lee Giles

Submitted by:
Professor David Casasent (Principal Investigator)
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213
(412) 268-2464

Date: March 1989

ABSTRACT

This represents the final report on our recent 4 year effort. We briefly review our prior feature space and distortion-invariant processors. We then highlight several of our optical AI processor concepts. Emphasis is given to new associative processor and directed graph optical systems for large knowledge base processing.

CHAPTER 1:

INTRODUCTION

This final report provides a comprehensive review of the past 4 years of our research. Chapter 2 summarizes various possible optical feature spaces that can be optically generated with emphasis on the use of computer generated holograms [1]. These operations demonstrate the growing flexibility and variety of operations possible on a fixed optical processor. Chapter 3 summarizes much of our distortion-invariant pattern recognition work [2]. This work has resulted in major and widely used techniques for 3-D aspect-invariant target recognition. In the course of years 2 and 3 of this program, we devised various optical AI processors. These include: a model-based processor [3,4], a symbolic processor [5,6], a rule-based processor [7], and a hierarchical processor [5], among others. Chapter 4 summarizes this work [8-11].

Our most recent efforts have concentrated on associative processors and directed graphs. Chapter 5 provides our optical directed graph large knowledge base initial concepts [12]. Chapter 6 includes our most recent directed graph large class work [13]. Chapters 7-10 detail our associative processor progress [14-17]. This includes attention to heteroassociative processors, new performance measures, improved encoding techniques, and storage capacity analyses of various associative processors (Chapter 7). It also involves new update techniques for adaptive and learning associative processors (Chapter 8). Our work also concerns the use of new key vector representation spaces such as feature spaces and symbolic data (Chapter 9), and initial ideas on new high capacity Ho-Kashyap associative processors for operation on linearly-dependent key vectors and improved performance (Chapter 10).

Chapter 11 lists all 82 papers published under this grant and all presentations and degrees granted under support of this AFOSR grant.

REFERENCES

1. D. Casasent, "Computer Generated Holograms in Pattern Recognition: A Review", Proc. SPIE, Vol. 532, pp. 106-118, January 1985.

2. D. Casasent and W.T. Chang, "Correlation Synthetic Discriminant Functions", Applied Optics, Vol. 25, pp. 2343-2350, 15 July 1986.

3. D. Casasent and S.A. Liebowitz, "Model-Based System for On-Line Affine Image Transformations", Proc. SPIE, Vol. 638, pp. 66-75, March-April 1986.

4. D. Casasent and S.A. Liebowitz, "Model-Based Knowledge-Based Optical Processors", Applied Optics, Vol. 26, pp. 1935-1942, 15 May 1987.

5. D. Casasent, "Optical AI Symbolic Correlators: Architecture and Filter Considerations", Proc. SPIE, Vol. 625, pp. 220-225, January 1986.

6. D. Casasent and E. Botha, "Knowledge in Optical Symbolic Pattern Recognition Processors", Optical Engineering, *Special Issue on Optical Computing and Nonlinear Optical Signal Processing*, Vol. 26, pp. 34-40, January 1987.

7. D. Casasent and A. Mahalanobis, "Rule-Based Symbolic Processor for Object Recognition", Applied Optics, Vol. 26, pp. 4795-4802, 15 November 1987.

8. D. Casasent, "Optical Pattern Recognition and Artificial Intelligence: A Review", Proc. SPIE, Vol. 754, pp. 2-11, January 1987.

9. D. Casasent, "Optical Pattern Recognition and AI Algorithms and Architectures for ATR and Computer Vision", Proc. SPIE, Vol. 755, pp. 83-93, January 1987.

10. D. Casasent, "Optical Artificial Intelligence Processors", *IOCC-1986 International Optical Computing Conference* (Israel), Proc. SPIE, Vol. 700, July 1986, pp. 246-250, 1986.

11. D. Casasent, "Scene Analysis Research: Optical Pattern Recognition and Artificial Intelligence", SPIE, *Advanced Institute Series on Hybrid and Optical Computers*, Vol. 634, pp. 439-456, Leesburg, Virginia, March 1986.

12. D. Casasent and E. Baranoski, "Directed Graph for Adaptive Organization and Learning of a Knowledge Base", Applied Optics, Vol. 27, pp. 534-540, 15 February 1988.

13. D. Casasent. S.D. Liu and H. Yoneyama, "Multiple Directed Graph Large-Class Multi-Spectral Processor", Proc. SPIE, Vol. 974, pp. 207-217, August 1988, San Diego.

14. D. Casasent and B. Telfer, "Key and Recollection Vector Effects on Heteroassociative Memory Performance", Applied Optics, Vol. 28, pp. 272-283, 15 January 1989.

15. B. Telfer and D. Casasent, "Updating Optical Pseudoinverse Associative Memories", Applied Optics, submitted June 1988.

16. D. Casasent and B. Telfer, "Optical Associative Processors for Visual Perception", Proc. SPIE, Vol. 882, pp. 47-59, January 1988.

17. B. Telfer and D. Casasent, "Ho-Kashyap Associative Processors", Proc. SPIE, Vol. 1005, November 1988.

CHAPTER 2:

"COMPUTER GENERATED HOLOGRAMS IN PATTERN RECOGNITION: A REVIEW"

CHAPTER 3:

"CORRELATION SYNTHETIC DISCRIM'NANT FUNCTIONS"

Computer generated holograms in pattern recognition: a review

David Casasent
Carnegie-Mellon University
Department of Electrical and
Computer Engineering
Pittsburgh, Pennsylvania 15213

Abstract. Holographic optical elements and computer-generated holograms (CGHs) offer many attractive properties and uses. In this review, we consider the use of holography in pattern recognition, with specific attention to recent work with CGH elements.

Subject terms: holography; chord distribution; computer-generated hologram; coordinate transformation; correlator; feature extractor; Fourier coefficient; holographic optical element; moment; optical pattern recognition; Sobel transform; space-variant transform.

Optical Engineering 24(5), 724-730 (September/October 1985).

CONTENTS

1. INTRODUCTION

Holography has many uses, roles, and applications in pattern recognition.[1-3] Several recent reviews and journal special issues address optical pattern recognition (OPR)[2] and holography in pattern recognition.[2-3] In this present review, we consider recent work with emphasis on holographic optical element (HOE) and computer-generated hologram (CGH) holographic elements for pattern recognition. The basic CGH and HOE concepts were initially advanced by Lohmann and Paris.[4] An excellent review of implementations by W. Lee exists,[5] together with the proceedings of a recent international SPIE conference.[6] It clearly notes the wide versatility of these elements and the considerable interests in their use. A particularly attractive use of CGH for pattern recognition, as well as for optical interconnections, is based on the original work of Bryngdahl in CGHs for coordinate mapping.[7-8] The kinoform phase CGH[64] is also most attractive for light efficiency. OPR covers many architectures, algorithms, and techniques. The details of many of the optical algorithms and architectures are provided in several recent reviews.[9-11] The associated digital realization of the basic feature extraction algorithms noted can be found in Ref. 12.

In Sec. 2, we review the basic coherent optical processing architectures. We restrict attention to coherent architectures in general since suitable sources for these systems are readily available and because of the large amount of work in this area. The classic optical Fourier transform (FT), image multiplication, and frequency plane correlator (FPC) architectures are discussed. The basic elements of a pattern recognition system are then presented (Sec. 3) in general form. Image processing (Sec. 4), feature space generation (Sec. 5), recent optical correlator research (Sec. 6), and new filter functions to provide more practical optical correlators and to greatly aid in fabrication of small-sized optical correlator architectures (Sec. 7) are then advanced. In Secs. 3 through 7, emphasis is given to the use of CGHs in the realizations of different OPR architectures, processing functions, and their fabrication. The examples chosen are those for which CGHs thus far have been demonstrated to have use.

Many of the image processing basics required (Sec. 4) are detailed in Ref. 13. Edge enhancement is a particularly necessary operation for multisensor and other object identification applications. These operations are best detailed in Ref. 14. The optical CGH architectures to achieve these functions are based on the work in Refs. 15 and 16 and follow the basic concepts first addressed in Ref. 4. The optically generated feature spaces (Sec. 5) include Fourier coefficients,[17-22] moments,[23-27] chord distributions,[28,29] and space-variant transform feature spaces.[36-41] The optical feature extraction architecture (Sec. 5) and concepts (Sec. 2) are reviewed elsewhere[11,30] and their optical CGH realization is fully detailed elsewhere.[14-41] The recent optical correlator research using CGHs (Sec. 6) follows Refs. 43 through 53 with the synthetic discrimination function correlator research in Sec. 7 based on Refs. 54 through 61. The fabricated architectures discussed are detailed in Refs. 42, 62, 63, and 65.

2. BASIC OPTICAL PROCESSING ARCHITECTURES

Many different optical architectures exist that are suitable for pattern recognition using CGHs. The basic coherent optical processor is the Fourier transform (FT) architecture (Fig. 1) that produces at P_2 the FT of the input pattern at P_1. The architecture of Fig. 9 (Sec. 5.3) is another useful general architecture that demonstrates several pattern recognition and data processing features of optical systems. Figure 9 shows plane P_1 imaged onto plane P_2. This yields the product of the 2-D distributions in each plane. The output lens in the system achieves the integration or point-by-point summation of these 2-D product data. Thus the integrated product of the two functions is formed at the output plane P_3. The classic frequency plane correlator[1] of Fig. 2 realizes the correlations of g and h at P_3 with g entered at P_1 and the conjugate transform H^* of h at the FT plane P_2 as a matched spatial filter (MSF). The output at P_3 is the 2-D correlation function

$$\mathcal{F}[FG^*] = \iint g(x,y)h(x-\tau_1, y-\tau_2)dxdy \ , \qquad (1)$$

Invited Paper HO-101 received Feb. 8, 1985; revised manuscript received March 29, 1985; accepted for publication March 29, 1985; received by Managing Editor May 31, 1985. This paper is a revision of Paper 532-10 which was presented at the SPIE conference on Holography: Critical Review of Technology, Jan. 24-25, 1985, Los Angeles, Calif. The paper presented there appears (unrefereed) in SPIE Proceedings Vol. 532.

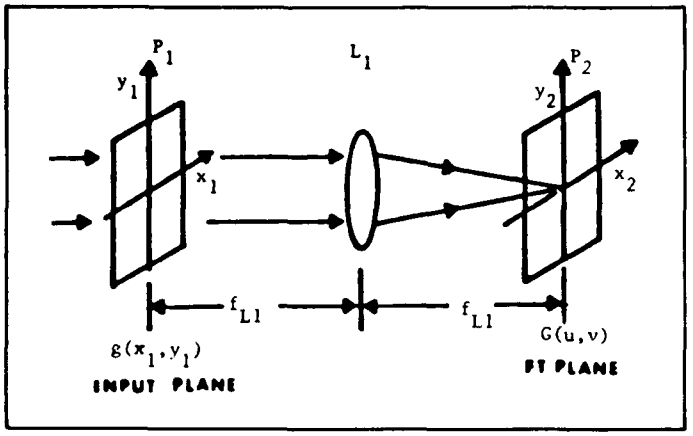


Fig. 1. Coherent optical Fourier transform processor.

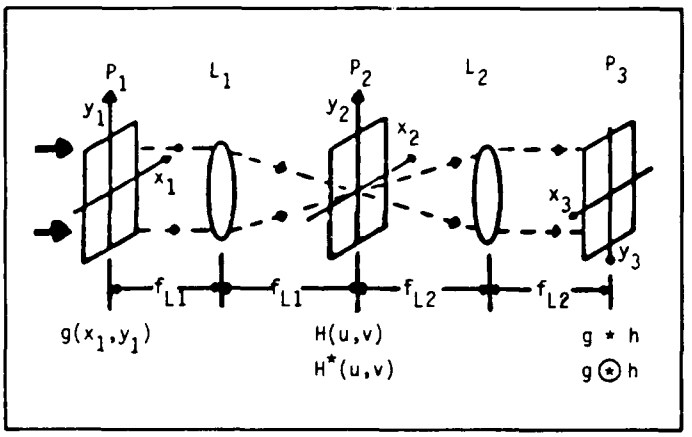


Fig. 2. Coherent optical matched spatial filter (MSF) frequency plane correlator.

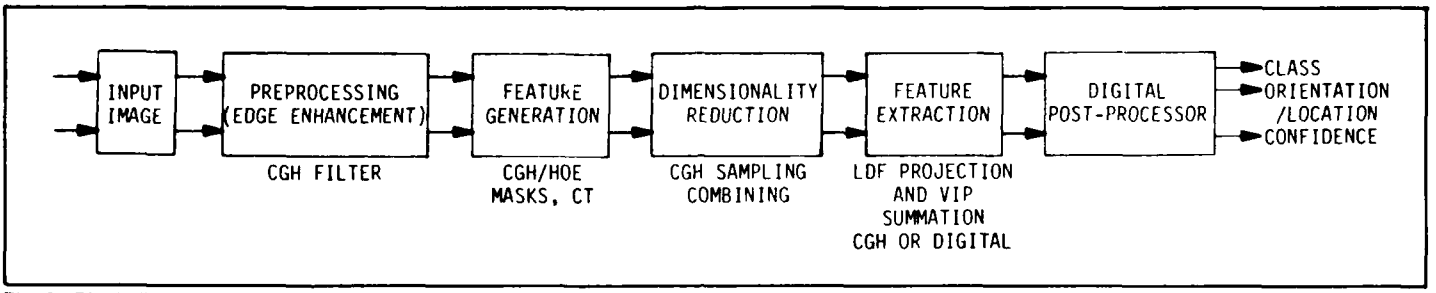


Fig. 3. Block diagram of a general feature extractor pattern recognition system with emphasis on the use of CGHs.

where $\mathcal{F}$ denotes the FT operator and τ_1 and τ_2 the correlation plane shift variables.

3. FEATURE EXTRACTION CONCEPTS

In Fig. 3, the basic feature extractor approach to pattern recognition is outlined in block diagram form. The blocks are selected to separately identify the different CGH and other units in the feature extraction pattern recognition systems to be discussed in subsequent sections. The CGH roles within each block are noted beneath the corresponding block. The preprocessor performs noise reduction, edge enhancement, and other functions by local operation such as median filtering, Sobel operators, etc., as detailed in Sec. 4. The feature generators (Sec. 5) produce scalars that describe the input image as the elements of a feature vector. Five different optical feature generators using CGHs are detailed in Sec. 5. Dimensionality reduction refers to reducing the dimensionality of the feature space used (to simplify post-processing requirements). An optical wedge ring detector (WRD) simulated by a CGH has considerable use in dimensionality reduction, as discussed in Sec. 5.1 in conjunction with a Fourier coefficient feature space. This WRD is also useful in the generation of a chord feature space.

The feature extractor portion of such a processor involves the projection of the feature vector $\mathbf{x}$ onto a linear discriminant function (LDF) $\mathbf{w}_i$ for class i object identification. From the vector inner product (VIP) $\mathbf{w}_i^T\mathbf{x}$ value compared to a threshold T, a decision on the object class is made. Optical feature extraction using CGHs is possible by two general techniques. In Fig. 4(a), a CGH in the feature space provides the projection of $\mathbf{x}$ onto the LDF vector $\mathbf{w}_i$ for class i with the projected vector inner product scalar outputs for each class i formed at different spatial locations in the output plane.[31] The second method [Fig. 4(b)][32-35] operates on the input image space directly. All major types of feature extractor algorithms, including unitary transformations, can and have been demonstrated on these systems. These include Fukunaga-Koontz, Foley-Samon, Fisher, least squares, and dominant eigenvector feature extractors. The

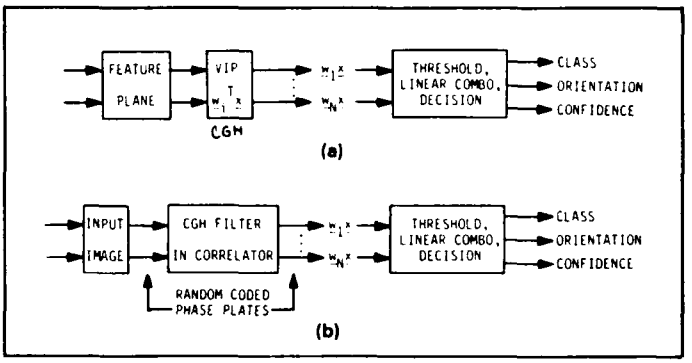


Fig. 4. Two approaches to the use of optical CGHs for feature extraction in (a) feature space and (b) image space.

details of these feature extractors in pattern recognition are provided elsewhere.[12]

4. CGHs IN IMAGE PROCESSING[13-16]

By the term image processing we generally refer to the production of a better image from the original input image. This is necessary with feature extractors to allow their efficient performance. The classic operations[13] include median filtering, edge enhancement, and various histogram operations. One must generally restrict preprocessing operations to such simple ones since they are easily realized in real time. Herein, we detail and demonstrate the use of CGHs and multiple MSFs to realize general local operators of any window size. We concentrate attention on edge enhancement operations since they are necessary in multisensor image pattern recognition and other similar cases.[13,14] One of the most attractive edge enhancement operators is the Sobel. This transforms an input image f into an edge-enhanced image g. It can be realized with f at P_1 and a CGH of a filter h at P_2 of Fig. 2. The P_3 output is $g = f * h$, where h is the sum of two 3×3 local point operators, as detailed elsewhere.[15] For the Sobel

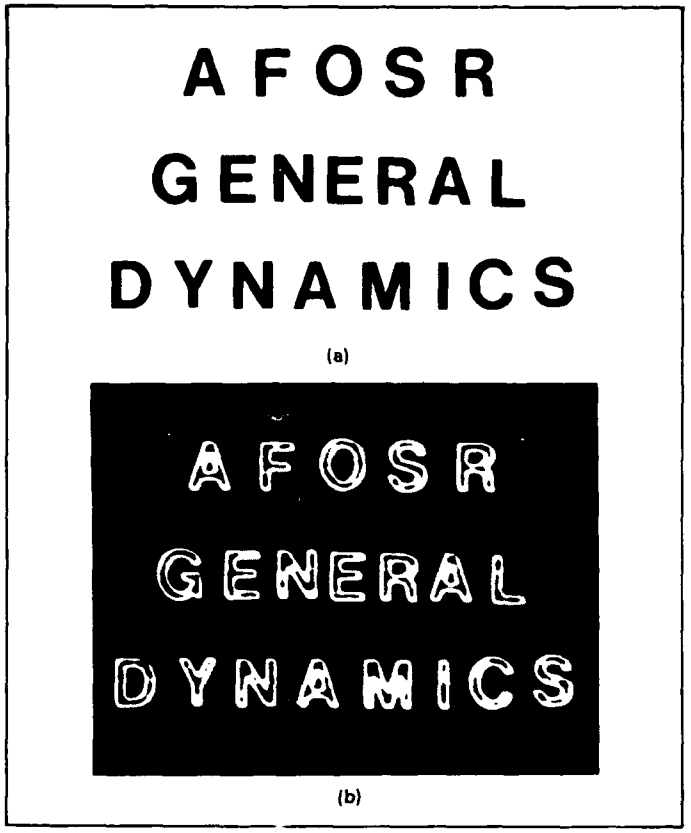


Fig. 5. CGH Sobel edge enhancement (b) of a binary input image (a).[16]

operator case, this filter function is the sum of a number of delta functions at different spatial locations and with different complex/bipolar weights. In Ref. 15, we detail how to realize these filters using multiple-exposed MSFs and using CGHs. The use of CGHs is superior to conventional holographic filters since they offer more flexibility in the size and type of local operators and in the efficiency and dynamic range of the filter function plus considerable ease of fabrication. They are also less susceptible to and more easily correctable for the t-E curve of the film used. Similar advantages apply to most uses of CGHs, specifically as the MSFs in an optical correlator. Figures 5 and 6 show examples of the optical Sobel edge enhancement of images with optically produced CGH filters.

5. CGH FOR FEATURE GENERATION

In all feature extractors, the generation of the features is the most computationally intensive operation. Herein, we detail four CGH optically generated feature space processors and provide references for the excellent performance possible with such algorithms.

5.1. CGH-produced wedge-ring sampled Fourier coefficient feature space[17-22]

The optical FT coefficient outputs at P_2 in Fig. 1 are the simplest optically generated features. This is also a most useful feature space since the radial FT pattern provides scale information, its angular distribution provides rotational information, and its magnitude is translationally invariant. The use of an HOE in place of the FT lens is obvious. An FT space is also attractive because it is well known to be very useful for data compression or dimensionality reduction. Sampling of the FT plane with a detector with wedge- and ring-shaped detector elements (a wedge-ring detector, WRD) is the most obvious technique[20] for dimensionality reduction that has been widely exploited.[21] In Ref. 17, we detail the optical synthesis of such WRDs using a CGH. The CGH system in Ref. 19 intended for fluid flow analysis can also achieve such a WRD synthesis, as can the techniques in Ref. 18 intended for fiber optic coupling applications.

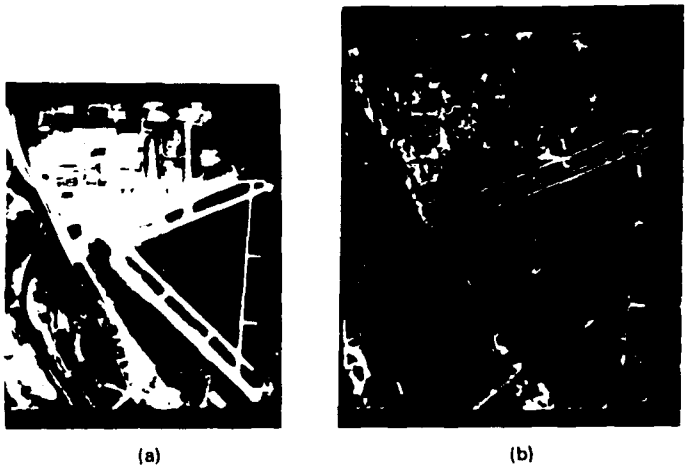


Fig. 6. CGH Sobel edge enhancement (b) of a gray-scale aerial image (a).[16]

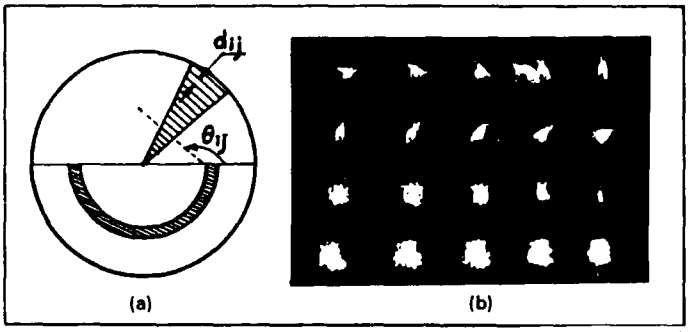


Fig. 7. CGH synthesis (a) of a wedge-ring detector sampling system and the corresponding output plane results (b).[17]

Reference 22 best details the use of such a feature space for pattern recognition and includes comparisons of the performance of various feature extractors. Figure 7(a) shows the CGH format to produce a CGH/WRD. (A grating with different spatial frequencies and angular orientations is present in each P_2 region.) Figure 7(b) shows the outputs (wedges on the top and rings on the bottom) from such a CGH after transformation and focusing onto the plane of a rectangular detector array. The design, fabrication, and initial tests on this WRD CGH are detailed in Ref. 17. The fabrication of such a system in a 2 in. $\times$4 in. volume has been noted,[17] and the basic concepts have been experimentally verified. Detailed use of this pattern recognition concept in the identification of letters and vehicles using different feature extractors has also been reported[22] with promising results.

5.2. CGH-produced chord distribution feature space[28,29,66]

For a binary boundary object, a chord of length r and angle θ can be drawn between *all pairs* of points on the boundary. The object can then be described by the chord transform $h(r,\theta)$ plot of the distribution of the length and angle of all chords. This distribution is attractive because $h(r)$ is invariant to in-plane object rotations and $h(\theta)$ is invariant to object scale changes. The autocorrelation of the input object is usually achieved optically (using a CGH point hologram, nonlinear crystals, the intensity of the Fourier transform, a joint transform correlator, etc.) and produces the distribution

$$h(d_x,d_y) = \iint g(x,y)g(x-d_x,y-d_y)dxdy \qquad (2)$$

of the horizontal and vertical projections d_x, d_y of all chords.[28] To reduce this space further for easier feature extraction, we suggested[28] WRD sampling of the autocorrelation. This is achieved with the

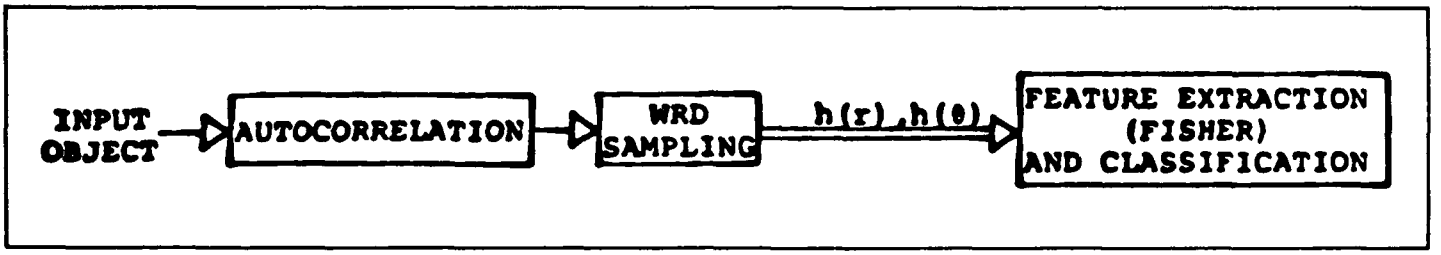


Fig. 8. Block diagram of an optical CGH chord distribution feature space pattern recognition system.

CGH WRD noted in Sec. 5.1. This produces separate length h(r) and angle h(θ) chord distributions.

We also[28,66] noted extensions of this generalized chord distribution for silhouette objects and gray-scaled objects. The optical synthesis technique achieves these general chord distributions at no increased cost. In both the FT and chord sampling WRD, the use of an optical CGH achieves all the necessary interpolation directly. (Such interpolation operations can be time-consuming digitally, and generally a table look-up method is required digitally.) Figure 8 shows the general block diagram for synthesis of such a chord distribution feature space. Reference 28 includes extensive simulation results on the pattern recognition performance of a chord distribution feature space for ship identification. (Perfect 100% correct classification is achieved for a two-class problem with 12 training set images per class in the face of 3-D out-of-plane rotations or aspect distortions with tests performed on 36 images per class.) The extension of this feature space to in-plane scale and rotational distortions and to the estimation of these orientation parameters of the object from this feature space were recently discussed.[66]

5.3. CGH-produced moment feature space[23–27,67]

The moments of an input object are a well-known and used feature space. The moments

$$m_{pq} = \iint f(x,y)\,x^p y^q dxdy \qquad (3)$$

of an input object f(x,y) can be calculated on the system of Fig. 9. This system produces on-axis at P_3

$$u_3(0,0) = \iint f(x,y)g(x,y)dxdy\ , \qquad (4)$$

as discussed in Sec. 2. With the mask function g being different monomials $x^p y^q$ on different spatial frequency carriers, the P_3 outputs are the moments in Eq. (3) all generated in parallel and all occurring at spatially separated locations in P_3 where they can be measured easily by separate detectors. This synthesis concept was initially advanced in Refs. 23 and 24. One can realize the mask g(x,y) required using cosine functions[24] or exponential functions[25] for the carriers with each being formed by CGHs. In the first case, only an amplitude CGH pattern is necessary. In the second case, a complex-valued CGH pattern is required.

The moments computed on the system of Fig. 9 are easily corrected for various mask resolution, system nonlinearity, and misfocusing errors.[24] A full two-level estimator, feature extractor, and post-processor architecture using a moment feature space have been detailed[26,27] and extensively tested on a large 324 image five-class data base of pipe parts,[26] on a large 180 image five-class ship data base,[27] on real image data, and with attention to precise distortion parameter estimation. All of these tests were performed with 3-D aspect view distortions present. The system of Fig. 9 with a cosine mask should be capable of optically producing the first 21 moments up to fifth order in parallel. An example[67] of ten optically generated moments using a complex-valued CGH exponential mask is shown in Fig. 10. Figure 10(a) shows the output P_3 plane format. (Note that the + and − parts of m_{01}, m_{10}, and m_{11} are generated separately.)

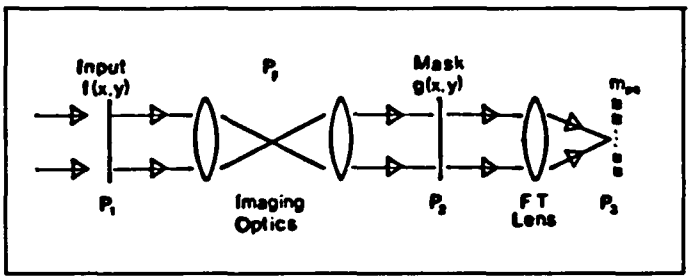


Fig. 9. Optical CGH system to produce a moment feature space in parallel.[24]

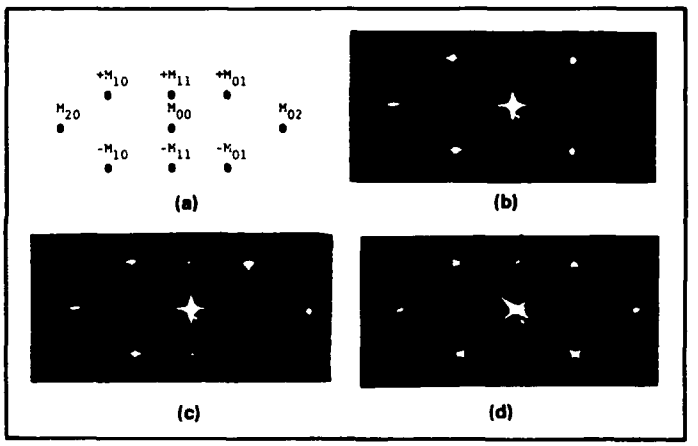


Fig. 10. Output format (a) and examples (b), (c), and (d) of optically produced moment features using CGHs.[67]

Figure 10(b) shows the P_3 output for a rectangular object with 2:1 aspect. From m_{20}^2/m_{02}^2, we calculate 2.02 as the aspect ratio, and thus this aspect estimate is quite accurate. Since the + and − parts of m_{10} and m_{01} are equal, we know the object's location (i.e., it is centered). In Fig. 10(c), the input object is shifted. The + and − parts of m_{01} change, and from these we can determine the object's shift. Figure 10(d) shows the result when the object is rotated. From the m_{11} and other values obtained, we can determine the amount of rotation of the input object.

5.4. Space-variant CGH feature space generation[36–41]

If the input object at f(x,y) is coordinate transformed, then the magnitude of its FT offers several interesting distortion-invariant feature spaces. This class of optical system is referred to as a space-variant optical processor. The most comprehensive review of this type of pattern recognition is in Ref. 36. If the object distortion can be mathematically described as an equation, then, using a general formula,[36] the required coordinate transform (such that the resultant FT pattern is invariant to this distortion) can be determined. If the dimension of the distortion space is greater than the dimensionality of the processor (e.g., two is the maximum conventionally available), multiple-pass techniques have been detailed. The distortion parameters of the object can be determined from the phase of the FT or

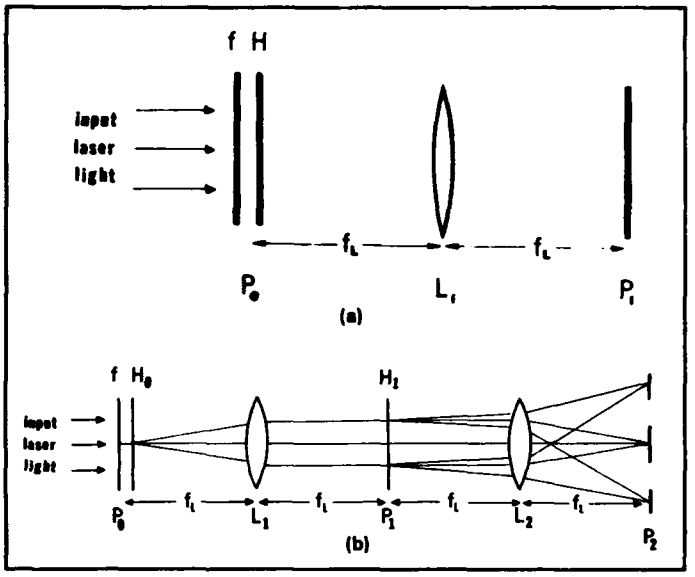


Fig. 11. Optical realization of coordinate transforms using CGHs to produce (a) the FT of the coordinate transformed function and (b) the image of the coordinate transformed function.

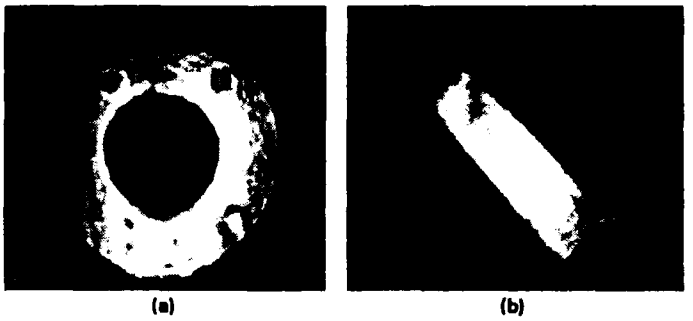


Fig. 12. Examples of experimental coordinate-transformed images produced using CGHs and the system of Fig. 11(b).[37,38]

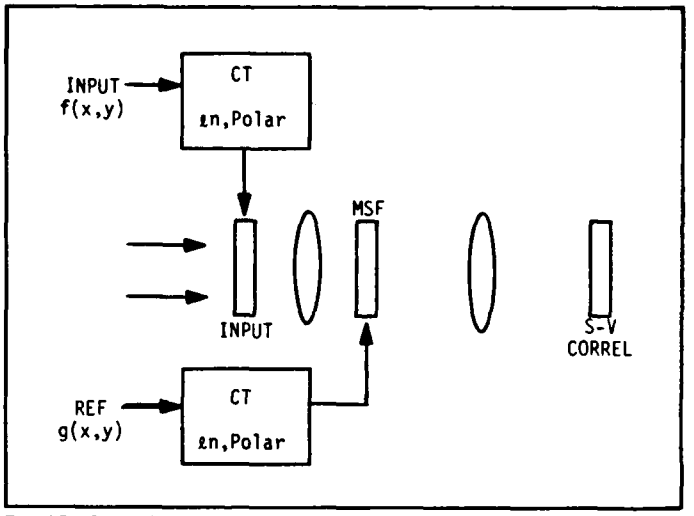


Fig. 13. Optical space-variant distortion-invariant correlator using CGHs to implement the coordinate transformation (CT).

from a space-variant correlation (Sec. 6). Here we concentrate on the realization of space-variant feature spaces using CGHs to produce the coordinate transform optically. (Since this is the most time-consuming and computationally intensive operation, attention to its optical realization is important.)

The most studied space-variant transformations are the scale-invariant Mellin transform using a ℓn coordinate transform (Refs. 37 and 38 detail the realization of the required CGH), a polar transform that is rotationally invariant,[41] a combined scale and rotation invariant transform,[36] and a coordinate transform for stereo perspective distortions. Much attention has recently been given to the realization of coordinate transforms with one[40] or with a cascade[39] of two CGHs and with attention to the use of faceted multiple-exposure holograms in different spatial locations. Here we review original work on CGH coordinate-transformed space-variant systems[37,38] that does not seem to be generally known. In Fig. 11, we show the system to realize the FT of a coordinate-transformed function [Fig. 11(a)] or an image plane representation of the coordinate-transformed function [Fig. 11(b)]. In Fig. 11(a), H is a CGH phase function $\exp[j\phi(x,y)]$. For a Mellin transform, $\phi(x,y) = x\ell nx - x + y\ell ny - y$. In Fig. 11(b), H_0 has a quadratic phase function (a lens or a linear frequency grating). In the first order at P_1, this produces an extended spectrum with geometric similarity to the input image (i.e., $x = u, y = v$) with light from different object points in P_0 now spatially separated. The second CGH H_1 placed at the first order in P_1 achieves the desired coordinate transform. For a Mellin transform, $\phi(u,v) = u\ell nu - u - 0.5u^2 + v\ell nu - v - 0.5v^2$. Figures 12(a) and 12(b) show the output P_3 patterns for the log-scaled $f(\exp\xi, \exp\eta)$ image plane representations of a donut and a tilted rectangular input object $f(x,y)$ using the CGH noted above. We have also produced the Mellin transform function with a single CGH[36] and have discussed realization of the polar transform using a CGH.[37] For feature extraction, the FT space for the coordinate-transformed function is most appropriate, since the magnitude of these coefficients (Mellin transform coefficients, etc.) are scale or rotationally invariant.

6. RECENT OPTICAL CORRELATOR RESEARCH USING CGHs

The use of CGHs has offered significant practical advantages for optical correlators and for their fabrication. Several examples of recent work in this area are now discussed.

During MSF synthesis of a reference function at P_2 of Fig. 2, one can use a converging reference beam. This forms a combined HOE/MSF at P_2, avoids the need for lens L_2, and results in simpler optical component positioning. This architecture has been discussed and experimentally demonstrated using a laser diode source.[42] An advanced version of this system using multiple laser diode sources and several space-multiplexed MSFs was researched, described, and initially demonstrated at Huntsville,[65] and a compact portable real-time system was fabricated and demonstrated by the Environmental Research Institute of Michigan (ERIM).[63] This significant advancement in optical correlator fabrication used HOEs rather than CGHs. Section 7 discusses a recent compact correlator using CGHs and advanced MSFs.

The use of space-variant processing in a correlator (Fig. 13) produces a distortion-invariant correlator with the location of the output correlation peak porportional to the distortion parameters. Demonstrations of such a system for scale, rotation, and scale/rotation invariance have been published. Such systems can be efficiently fabricated using CGHs to implement the coordinate transform operation (Sec. 5.4) and for fabrication of the MSF.

As optical correlators are now becoming more practical, recent attention has again been given to the importance of optical efficiency of the MSF and the light budget of such optical correlators.[43] If the MSF is synthesized using a CGH, the advantages of improved flexibility, better linearity, and more flexible synthesis procedures, as well as higher efficiency using phase filters, are possible.[45-48] Similarly, CGHs can be used to improve noncoherent correlators (however, shift-invariance can only be achieved at a significant increase in space bandwidth product).[49,50] An attractive laboratory experiment worth noting is the recent use of a Litton magneto-optic device (MOD) as a real-time spatial light modulator for correlation using a CGH. An example of this is shown in Fig. 14, where the input pattern, the binary CGH, and the output correlation plane pattern are shown.

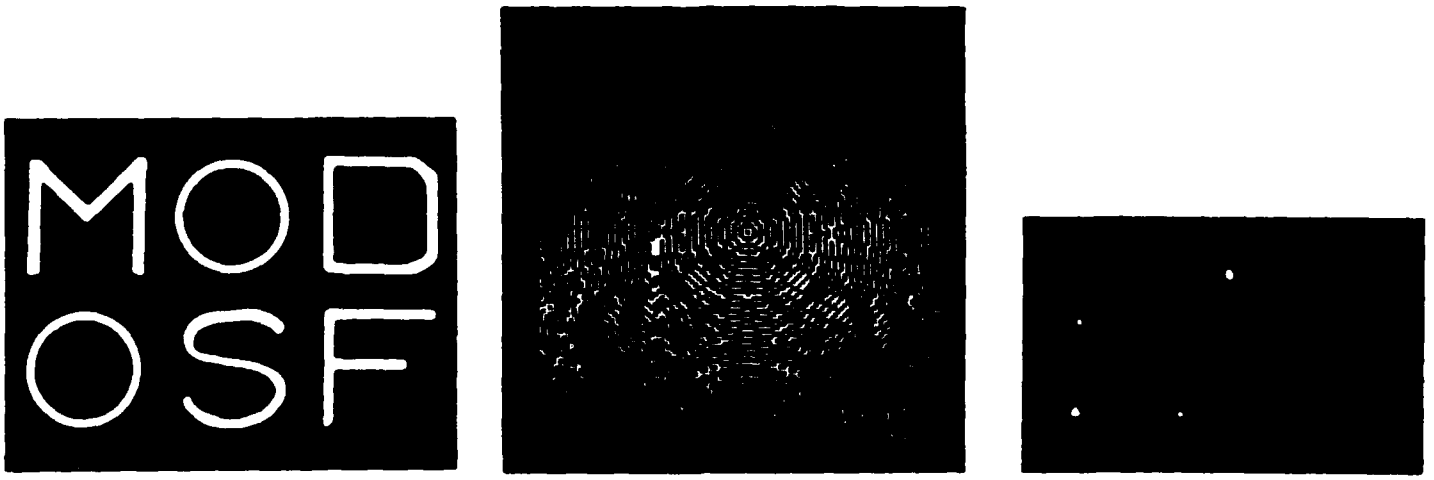


Fig. 14. Real-time optical binary correlation with a CGH of the letter O using the Litton MOD: (a) input image, (b) CGH, (c) output correlation plane pattern.[52]

The CGH was formed to recognize the letter O, and peaks are apparent in the output at the location of all O's in the input image. The binary nature of the MOD required a binary CGH to be used. The space bandwidth product of the present device is low (128×128 for the demonstration shown), and its light transmittance is presently only several percent. However, research in these areas is now in progress.

7. DISTORTION-INVARIANT OPTICAL CORRELATORS

The major shortcoming of the classic optical correlator is its lack of distortion invariance. One can synthesize a weighted MSF,[69,70] and CGHs allow better control of these or any MSFs. This can reduce the sensitivity of an MSF to within-class object distortions. One technique for full distortion invariance is to use multiple MSFs at spatially different locations in P_2 (each being an MSF of the reference object with a different scale, rotation, or aspect view). The differences between these views are determined by the sensitivity of the weighted MSF used. To access these multiple MSFs in parallel, the FT of the input object must be replicated. A CGH of multiple delta functions can achieve this. The architecture shown in Fig. 15 achieves this. The output is the correlation of the input with all references superimposed. Researchers at Huntsville[65] and Grumman[68] have used this technique in several optical MSF correlators.

The most attractive recent approach to a 3-D distortion-invariant optical correlator uses a synthetic discriminant function (SDF) filter at P_2 of Fig. 2. This SDF is a linear combination of several training set images of the objects to be identified or rejected. It can be designed for intraclass recognition and two-class to multiclass recognition.[54] These filters have provided excellent initial results on large ship[55] and automatic target recognition (ATR) target[56] test data.[57] The attractive performance of SDF correlators has caused much interest in them and an entire recent conference session on their realization using phase-only filters and CGHs.[61] Riggins and Butler simulated a projection intraclass SDF using a CGH[58] and distinguished between several types of SDF CGHs such as phase-only, bleached, etc.[59] Gianino and Horner[44] noted that phase-only filters appear to be more sensitive to distortions. Thus, the use of SDF phase-only CGHs should be a quite attractive implementation. Initial tests[60] seem to yield good results. However, all of the above tests have been performed only on limited data and not using the newest correlation filters, and have not yet considered intraclass recognition and discrimination. At Carnegie-Mellon University (CMU), recent tests were performed with phase-only SDFs, SDFs with various quantized amplitude and phase levels, and other CGH SDFs. With the proper number of amplitude and phase levels (a very low number), we have obtained excellent SDF CGH correlation results, excellent performance on real imagery, excellent discrimination, and intraclass recognition on large data bases. Thus, an SDF-based optical correlator appears to provide an excellent solution to distortion-invariant, multiclass, shift-invariant object pattern recognition in clutter.

Recently, an advanced and most compact real-time optical correlator was fabricated by General Dynamics-Pomona. The system is shown in Fig. 16. It is less than 5 in. in diameter and approximately 12 in. long and is thus suitable for use in a 5 in. missile. It is a real-time processor presently using a liquid crystal input light valve. It employs advanced correlation SDFs to achieve multiclass 3-D distortion-invariant pattern recognition. Multiple CGHs of SDF MSFs are recorded in the system. The system has performed well in initial tests. It is scheduled for tower tests and captive helicopter tests in 1985. This advancement is presently the most powerful and most compact optical correlator produced. It unifies the major research in optical pattern recognition, CGHs, correlators, and fabrication techniques. The use of SDFs allows distortion-invariant operation The use of CGHs and SDFs provides flexibility together with fabrication in a compact system with low size, weight, cost, and power dissipation.

8. SUMMARY AND CONCLUSION

Computer-generated holograms have been shown to have many attractive possibilities for pattern recognition. The proper use of CGHs appears to be for operations not easy with conventional optics, for maintaining small size and cost, as well as for use in performing the more computationally intensive operations required in a given pattern recognition algorithm. As we have shown and detailed, CGHs allow many operations and functions required in pattern recognition to be performed faster and in a system of lower cost, size, volume, and power dissipation than other technologies can provide. The CGH feature extraction pattern recognition methods discussed included special detection/sampling (e.g., a wedge-ring detector), coordinate transformation operations for space-variant processors, nonlinear local operations for image preprocessing, parallel feature extraction and transformation, chord distribution generation, and as the monomial masks required for a moment feature space generation. CGHs also have many attractive uses in correlators. CGHs permit flexible MSF synthesis with many necessary weighting parameters quite easily variable and with phase filters possible to yield high efficiency systems. CGHs can also be used for object replication with multiple matched spatial filters and for the realization of synthetic discriminant functions. CGHs and synthetic discriminant functions have made fabrication and realization of practical distortion-invariant optical correlators most attractive. A

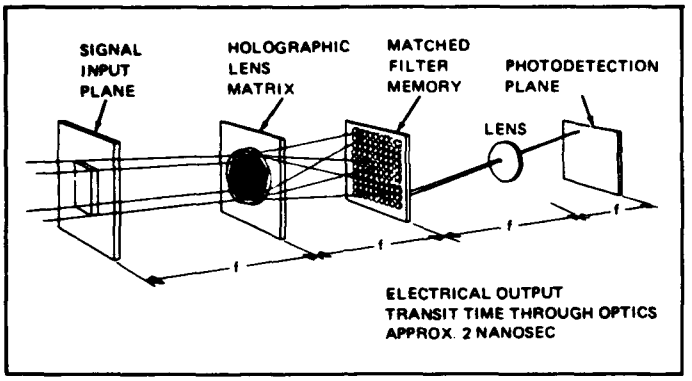


Fig. 15. Optical correlator with space-multiplexed MSFs using a CGH point hologram to replicate the Fourier transform of the input object.[68]

powerful distortion-invariant correlator in a 5 in. package suitable for missile guidance is now possible and under testing.

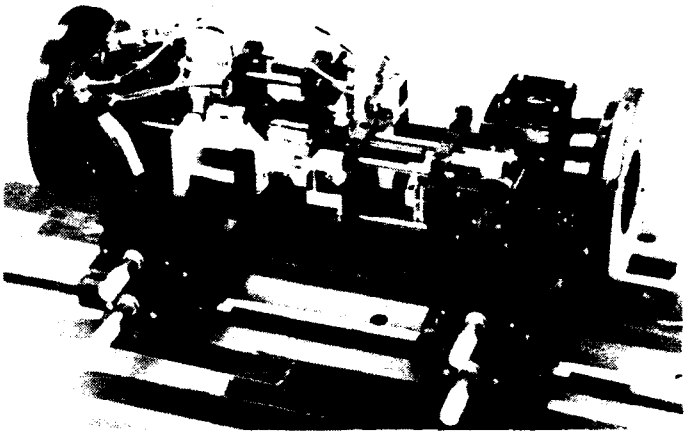


Fig. 16. Phtograph of a compact 5 in. real-time SDF/CGH-based multichannel correlator (courtesy D. Fetterly, General Dynamics-Pomona).

9. ACKNOWLEDGMENTS

The resarch support of the Air Force Office of Scientific Research (Grant AFOSR-84-0293 and Agreement F49620-83-C-0100) for the CMU research included herein is gratefully acknowledged, as are the Westinghouse-funded facilities of the CMU Center for Excellence in Optical Data Processing used in the various CMU experiments discussed.

10. REFERENCES

1. A. VanderLugt, IEEE Trans. Info. Theory IT-10, 139 (1964).
2. Joseph L. Horner, ed., Optical Engineering Special Issue on Optical Pattern Recognition, Vol. 23(6), pp. 687–747 (Nov./Dec. 1984).
3. D. Casasent, in *Industrial and Commercial Applications of Holography*, M. Chang, ed., Proc. SPIE 353, 6 (1983).
4. A. Lohmann and D. Paris, Appl. Opt. 7, 651 (1968).
5. W. H. Lee, in *Progress in Optics*, Vol. XVI, E. Wolf, ed., pp. 121-232, North-Holland, New York (1978).
6. Sing H. Lee, ed., *International Conference on Computer-generated Holography*, Proc. SPIE Vol. 437 (1983).
7. O. Bryngdahl, J. Opt. Soc. Am. 64, 1092 (1974).
8. O. Bryngdahl, Opt. Commun. 10, 164 (1974).
9. D. Casasent, IEEE Spectrum, p. 28 (March 1981).
10. D. Casasent, Opt. Eng. 24(1), 26 (1985).
11. D. Casasent, in *Digital Image Processing*, A. G. Tescher, ed., Proc. SPIE 528, 64 (1985).
12. R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York (1973).
13. W. Pratt, IEEE Trans. Aeros. Electron. Sys. AES-10, 353 (1974).
14. Y. Barniv and D. Casasent, in *Processing of Images and Data from Optical Sensors*, W. H. Carter, ed., Proc. SPIE 292, 160 (1981). Also see Y. Barniv, H. Mostafavi, and D. Casasent, in *Image Processing for Missile Guidance*, T. F. Wiener, ed., Proc. SPIE 238, 156 (1980).
15. D. Casasent and J. Chen, Appl. Opt. 22, 808 (1983).
16. J. Chen and D. Casasent, Lasers and Electro-Optics 34, 24 (1982).
17. D. Casasent and J. Z. Song, in *Applications of Holography*, L. Huff, ed., Proc. SPIE 523, 227 (1985).
18. J. N. Cederquist and A. M. Tai, in *International Conference on Computer-generated Holography*, Sing H. Lee, ed., Proc. SPIE 437, 101 (1983).
19. R. Sandstrom and S. H. Lee, in *International Conference on Computer-generated Holography*, Sing H. Lee, ed., Proc. SPIE 437, 64 (1983).
20. G. G. Lendaris and G. L. Stanley, Proc. IEEE 58(2), 198 (1979).
21. H. Kasden and D. Mead, Proc. Elec. Opt. Sys. Des. 248, (1976).
22. D. Casasent and V. Sharma, Opt. Eng. 23(5), 492 (1984).
23. D. Casasent, J. Pauly, and D. Fetterly, in *Infrared Technology for Target Detection and Classification*, P. M. Narendra, ed., Proc. SPIE 302, 126 (1982).
24. D. Casasent, R. L. Cheatham, and D. Fetterly, Appl. Opt. 21, 3292 (1982).
25. J. Blodgett et al., Opt. Lett. 7, 7 (1982).
26. D. Casasent and R. L. Cheatham, Proc. ASME, pp. 1-6 (Aug. 1984).
27. D. Casasent and R. L. Cheatham, in *Applications of Digital Image Processing VII*, Proc. SPIE 504, 19 (1984).
28. D. Casasent and W. T. Chang, Appl. Opt. 22, 2087 (1983).
29. D. J. H. Moore and D. J. Parker, Pattern Recognition 6, 149 (1974).
30. D. Casasent, in *Real Time Signal Processing VI*, Keith Bromley, ed., Proc. SPIE 431, 263 (1983).
31. D. Casasent and H. Okuyama, "High-Dimensionality Feature-Space Processing with Computer Generated Holograms," presented at SPIE conference on Intelligent Robots and Computer Vision, Sept. 16-20, 1985, Cambridge, Mass. To be published in Proc. SPIE Vol. 579.
32. J. R. Leger and S. H. Lee, Appl. Opt. 21, 274 (1982).
33. J. R. Leger and S. H. Lee, J. Opt. Soc. Am. 72, 556 (1982).
34. Z. H. Gu, J. R. Leger, and S. H. Lee, J. Opt. Soc. Am. 72, 787 (1982).
35. Z. H. Gu and S. H. Lee, Appl. Opt. 23, 822 (1984).
36. D. Casasent and D. Psaltis, in *Progress in Optics*, Vol. XVI, E. Wolf, ed., pp. 291-356, North-Holland, New York (1979).
37. D. Casasent and C. Szczutkowski, in *Optical Information Processing*, D. Casasent and A. A. Sawchuk, eds., Proc. SPIE 83, 91 (1977).
38. D. Casasent and C. Szczutkowski, Opt. Commun. 19, 217 (1976).
39. H. Bartelt and S. K. Case, Opt. Eng. 22(4), 497 (1983).
40. P. Haugen, H. Bartelt, and S. Case, Appl. Opt. 22, 2822 (1983); S. Case, P. Haugen, and O. Lokberg, Appl. Opt. 20, 2670 (1981). See also S. Case and P. Haugen, Opt. Eng. 21(2), 352 (1982).
41. D. Casasent and M. Kraus, Appl. Opt. 17, 1559 (1978).
42. F. Caimi, D. Casasent, M. Shen, and B. Feng, Appl. Opt. 19, 2653 (1980).
43. J. Horner and P. Gianino, Appl. Opt. 23, 812 (1984).
44. P. Gianino and J. Horner, Opt. Eng. 23(6), 695 (1984).
45. G. Indebetouw, T. Tschudi, and G. Herziger, Appl. Opt. 15, 516 (1976).
46. G. Indebetouw, Appl. Opt. 16, 1944 (1977).
47. G. Indebetouw, T. Tschudi, and J. Steffen, Appl. Opt. 17, 911 (1978).
48. T. Tschudi, in *International Conference on Computer-generated Holography*, S. H. Lee, ed., Proc. SPIE 437, 176 (1983).
49. A. Lohmann and C. Thum, Appl. Opt. 23, 1503 (1984).
50. H. Bartelt, Appl. Opt. 23, 1499 (1984).
51. W. E. Ross, D. Psaltis, and R. H. Anderson, Opt. Eng. 22(4), 485 (1983).
52. D. Psaltis, E. G. Paek, and S. S. Venkatesh, Opt. Eng. 23(6), 698 (1984).
53. F. Yu, X. Lu, and M. Cao, Appl. Opt. 23, 4100 (1984).
54. D. Casasent, Appl. Opt. 23, 1620 (1984).
55. D. Casasent, W. Rozzi, and D. Fetterly, Opt. Eng. 23(6), 716 (1984).
56. W. T. Chang, D. Casasent, and D. Fetterly, in *Processing and Display of Three-Dimensional Data II*, J. J. Pearson, ed., Proc. SPIE 507, 9 (1984).
57. D. Casasent and W. Rozzi, "Correlation Synthetic Discriminant Functions (SDFs) for Object Recognition and Classification in High Clutter, presented at SPIE conference on Applications of Digital Image Processing VII, Aug. 20-22, 1985, San Diego, Calif. To be published in Proc. SPIE Vol. 575.
58. J. Riggins and S. Butler, Opt. Eng. 23(6), 721 (1984).
59. S. Butler and J. Riggins, in *Analog Optical Processing and Computing*, H. John Caulfield, ed., Proc. SPIE 519, 78 (1985).
60. J. Horner and P. Gianino, in *Analog Optical Processing and Computing*, Proc. SPIE 519, 70 (1985).
61. H. John Caulfield, ed., *Analog Optical Processing and Computing*, Proc. SPIE Vol. 519 (1985).
62. K. G. Leib et al., Appl. Opt. 17, 2892 (1978).
63. J. Upatnieks, Appl. Opt. 22, 2798 (1983).
64. L. Lesem, P. Hirsch, and J. Jordan, IBM J. Res. Dev., pp. 150-155 (March 1969).
65. B. D. Guenther, C. R. Christensen, and J. Upatnieks, IEEE J. Quantum Electron. QE-15, 1348 (1979).
66. W. T. Chang and D. Casasent, in *Intelligent Robots and Computer Vision*, D. P. Casasent and E. L. Hall, eds., Proc. SPIE 521, 2 (1985).
67. R. A. Athale, C. L. Giles, and J. A. Blodgett, in *International Conference on Computer-generated Holography*, S. H. Lee, ed., Proc. SPIE 437, 48 (1983).
68. A. Reich et al., "High Accuracy Cruise Missile Terminal Guidance System," Final Report DAAK40-77-C-0089, Grumman Aerospace Corporation (1978).
69. D. Casasent and A. Furman, Appl. Opt. 16, 1652 (1977).
70. D. Casasent and A. Furman, Appl. Opt. 16, 1662 (1977).

Reprinted from Applied Optics, Vol. 25, Page 2343, July 15, 1986

Correlation synthetic discriminant functions

David Casasent and Wen-Thong Chang

Advanced filters are described for distortion-invariant space-invariant object identification and location in clutter using correlators. These correlation synthetic discriminant functions (SDFs) are extensions of earlier projection SDFs. They provide control of the sidelobe levels and the shape of the output correlation function as well as its peak intensity. The theory for synthesis of three such SDFs and a discussion of correlation plane detection criteria for use with these filters are presented.

I. Introduction

Correlators are one of the most powerful techniques for locating multiple objects in clutter without the need for segmentation. The realization of correlators by optical systems is obvious and well documented[1,2] as is the shift-invariant and processing gain features of such processors. In the presence of white Gaussian noise, correlators with a matched spatial filter (MSF) of the reference object being searched for are optimal. However, they also perform well in colored noise or structured clutter, as recently quantified.[3] Their major shortcoming has been their poor sensitivity to geometrical distortions between the input and reference object. The use of various feature spaces (geometrical moments, Mellin coefficients, chord distributions, sampled Fourier coefficients) in conjunction with various linear discriminant function feature extractors has been the major approach taken to achieve distortion-invariant pattern recognition.[2] However, segmentation and noise reduction are necessary preprocessing functions before feature extraction can be applied.

Thus, in this paper, we discuss a method to achieve distortion-invariant recognition while retaining the shift-invariant and processing gain advantages of correlators. The optical processing community has led and pioneered such research. However, many optical systems such as the space-invariant[4] and coded-phase[5] correlators are not shift-invariant and cannot handle multiple objects. Synthesis of the MSF in an optical correlator from a projection SDF[6] algorithm has been shown to achieve distortion invariance and retain shift invariance, but this filter cannot prevent large sidelobe levels from occurring in the correlation plane for the case of false (or true) targets.

In this paper, we review projection SDF synthesis and demonstrate its shortcomings by example (Sec. II). Synthesis of a correlation SDF which provides control of the shape of the correlation function is then detailed in Sec. III. In Sec. IV, examples of the superior correlation plane output response of these new SDFs are provided, and various correlation plane classification measures are discussed.

II. Projection SDFs[6]

We can state the requirements for a two-class pattern recognition system to be the recognition of objects $\{f\}$ in one class (true targets) and the discrimination of objects $\{g\}$ in a second class (false targets) in terms of the inner product operator $\langle\ \rangle$ as

$$\langle f \cdot h \rangle = 1 \text{ (true targets)}, \tag{1a}$$

$$\langle g \cdot h \rangle = 0 \text{ (false targets)}. \tag{1b}$$

The filter function h is required to satisfy Eq. (1a) for any input function that is a member of class 1 and to satisfy condition (1b) for any input function that is a member of image class 2. This corresponds to the recognition of various distorted versions of f (members of the set $\{f\}$) and rejection or discrimination of all distorted views of a second class of object $\{g\}$. This situation arises when the object classes $\{f\}$ and $\{g\}$ are similar. The distorted object views included in these two data sets and the application determine the specific geometrical distortions for which the resultant filter function h is invariant.

The images and filter function h are all 2-D image functions of the coordinates (x,y), but their spatial variables will generally be suppressed for simplicity in our system and filter synthesis description. Other constant constraints besides the 1 and 0 values noted in Eqs. (1) can be employed with no loss of generalization. To obtain a specific solution h to Eqs. (1), we

When this work was done both authors were with Carnegie-Mellon University, Department of Electrical & Computer Engineering, Pittsburgh, Pennsylvania 15213; W. T. Chang is now with Eastman Kodak Research Laboratories, Rochester, New York 14650.

Received 2 December 1985.
0003-6935/86/142343-08$02.00/0.

restrict h to be a linear combination of the training set images $\{f(x,y)\}$ and $\{g(x,y)\}$ or an orthonormal set of 2-D basis functions $\phi(x,y)$, i.e.,

$$h = \Sigma b_n f_n + \Sigma b_n g_n \tag{2a}$$

or

$$h = \Sigma a_n \phi_n. \tag{2b}$$

The basis functions $\{\phi\}$ are linear combinations of all $\{f\}$ and $\{g\}$ training set images and can be obtained by Karhunen-Loeve or Gram-Schmidt techniques.[7] Both filter descriptions in Eqs. (2) are equivalent. Equation (2a) is used in the synthesis of h from the training set images, and Eq. (2b) is attractive from general pattern recognition theory.

To solve for h given a training set of N_1 images $\{f\}$ and N_2 images $\{g\}$ in class 2, we must determine the coefficients a_n or b_n in Eqs. (2). We can achieve this by solving for the coefficients a_n as the vector solution **a** to[6]

$$\mathbf{Ta} = \mathbf{u}_1 = [\underbrace{1 \cdots 1}_{N_1}, \underbrace{0 \cdots 0}_{N_2}]^T, \tag{3a}$$

where **T** is the target matrix (its elements are the coefficients of each image in terms of ϕ_n), **a** is a vector whose elements are a_n coefficients in Eq. (2b), and **u** is the control vector with N_1 ones and N_2 zeroes. The choice of **u** is the key to SDF synthesis. With the first N_1 images members of $\{f\}$ and the last N_2 images members of $\{g\}$, the given **u** forces the output for all N_1 images $\{f\}$ to be 1 and the output for all N_2 images $\{g\}$ to be 0 as required by Eq. (1). The filter in Eq. (2a) can be similarly described by the coefficient vector solution **b** to[6]

$$\mathbf{Vb} = \mathbf{u}_1 = [\underbrace{1 \cdots 1}_{N_1}, \underbrace{0 \cdots 0}_{N_2}]^T, \tag{3b}$$

where **V** is the $N_1 + N_2$ dimensionality vector inner product (VIP) matrix of the training set data. (Its elements are the VIPs $\mathbf{f}_i^T\mathbf{f}_j$, $\mathbf{f}_i^T\mathbf{g}_j$, etc. of the training set with each image represented as an $N^2 = N \times N$ element vector, where N^2 denotes the number of pixels in the input image.) Many other variants beyond the SDF filter in Eqs. (1) and (2) exist.[6] These include the use of multiple levels in the **u** vector and the use of several filters. We consider only the two-class filter function solution in our present work. Extensions to other projection SDFs follow directly.

These filters perform quite well with central output peak values close to unity for class 1 objects and close to zero values for class 2 objects. However, the filter requirements in Eq. (1) only restrict the projection values (the correlation plane value at the central point) of an MSF of this SDF. When these filters are formed and used in an optical correlator this can present problems. This system is still shift invariant and hence provides the proper correlation plane values at the center of the target regardless of the target's location. For true targets, the largest correlation plane value occurs in the center of the target and hence the condition in Eq. (1a) is adequate. However, there is no restriction on the response of $h(x,y)$ to false targets $\{g\}$ at locations away from the central value of the false target correlation. Sidelobes in the correlation output for false targets can thus occur and achieve any level. These false targets sidelobe correlation levels can easily exceed any threshold devised for the central correlation peak value (as obtained with the present training set synthesis methods). Thus, for false targets with intensity or modulation variations much larger than those present in the training set, we observed correlation peaks displaced from the central correlation plane value that were above our threshold. Similar problems with sidelobe peaks in the correlation plane can arise for the case of true targets. However, such problems are generally of less concern.

In some sets of imagery, projection SDFs yield adequate response (i.e., no peaks above the threshold T_t) occurring anywhere on a false target. However, in general, we cannot guarantee such performance. Figure 1 shows an example of this. A projection SDF was designed to produce 1 outputs for all class 1 objects $\{f\}$ and 0 outputs for all class 2 objects. Twelve objects per class were used to synthesize this filter ($N_1 = N_2 = 12$). All images were different aspect views of the objects from a 20° depression angle. The twelve training images per class were each different aspect views 30° apart. The correlation plane response for a class 1 object (not in the training set) is shown in Fig. 1(a). This is typical of all similar response data. The peak value for the true class filter in Fig. 1(a) shows a peak

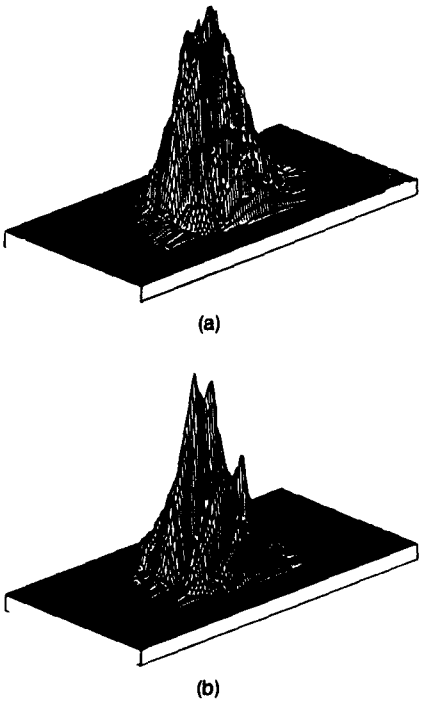


Fig. 1. True (a) and false (b) class correlation plane outputs for a projection SDF.

correlation plane value in the central pixel location (65,33) of 0.93. The largest peak value anywhere in this plane was 1.09 at location (66,41), which was displaced slightly from this central location (by 8 pixels). Thus the proper output results were as expected. A peak of ~1.0 (although shifted somewhat from the exact central location of the object) was obtained with this filter (even in the face of geometric distortions tested on imagery or object orientations not contained in the training set). In Fig. 1(b), we show the correlation plane response for a class 2 nontraining set object. Its central value at pixel location (65,33) is 0.1 (as desired, this is nearly 0.0, as specified by the filter). This specific central correlation plane value is not easily visible in the pseudo 3-D correlation plane view shown, but it is clearly evident from this plot that the central region of the correlation plane output has a low value. However, the largest peak value in the correlation plane is quite large (0.98) at pixel location (68,39). Large values thus exist around the central correlation peak for false targets as shown. These results are typical for many image data base cross-correlations for the target considered in these tests. Clearly, the full correlation plane response of this filter is poor for false class objects, and advanced versions of this filter function are needed to address such problems. We address these issues in Sec. III.

III. Correlation SDF Synthesis

We chose to control the correlation plane sidelobe response for false targets (and true targets) by modifying the filter's requirements by specifying the correlation plane outputs at more than just the central location of the target in the correlation plane. To achieve this and retain the simple VIP matrix formulation in Eq. (3b), we increase the size of our image training set to include shifted versions of each training set image. We denote the shifted image versions for the true class object by $\{f'\}$ and for the false target by $\{g'\}$. These images are shifted by d_s pixels with respect to the unshifted $\{f\}$ and $\{g\}$ training set images. Our notation is summarized in Table I. For our specific tests, we generally employed $N_1 = N_2 = 6$ training set images per object class with pixel shifts chosen as $d_s = \pm 5$ pixels horizontally and vertically. This corresponds to reference image pixel shifts of $(\pm d_s,0)$ and $(0,\pm d_s)$ pixels. Thus $N_s - 1 = 4$ shifted versions of each training set image or $N_s = 5$ total shifted and nonshifted training set images per class are used in correlation SDF synthesis. We retain our VIP matrix synthesis formulation by restricting the training set of data to include shifted and nonshifted versions of all imagery. We restrict the VIP projection output for shifted images to be zero or small for both the true and false class shifted training set imagery. This is equivalent to requiring the value of the correlation plane output (at $\pm d_s$ pixel shifts in x and y from the center of the target) to be zero. We discuss this further. We refer to these filters as correlation SDFs.

Table I. Notation and Parameters Associated with Correlation SDF Synthesis

f	Object in the class to be recognized (true target)
f'	Shifted version of f
g	Object in the class to be rejected (false target)
g'	Shifted version of g
d_s	Amount of shift in pixels
N_1	Number of training set images $\{f\}$ in class 1
N_2	Number of training set images $\{g\}$ in class 2
$N_s - 1$	Number of shifted versions of each image
$N_T = N_s(N_1 + N_2)$	Total training set size
h	MSF SDF filter function

A. Exact Correlation SDF-1 Synthesis

The filter h requirements in Eq. (1) for the case of correlation SDFs thus becomes

$$\langle f \cdot h \rangle = 1, \tag{4a}$$

$$\langle f' \cdot h \rangle = 0, \tag{4b}$$

$$\langle g \cdot h \rangle = 0, \tag{4c}$$

$$\langle g' \cdot h \rangle = 0. \tag{4d}$$

Conditions (4a) and (4c) are equivalent to Eqs. (1a) and (1b) as before. The new conditions (4b) and (4d) restrict the correlation value $\pm d_s$ pixels from the central correlation point to be zero. Let us now discuss the correlation plane pattern these filter requirements in Eqs. (4) yield. For a true class 1 target $\{f\}$, this yields a well-shaped correlation peak with a value of unity at the center of the object and 0 value d_s pixels away in $\pm x$ and $\pm y$. For a false class 2 objects, this filter synthesis produces zero values at the center of the object and $\pm d_s$ pixels away horizontally and vertically. For the true class 1 objects, this yields a well-shaped correlation peak. Suppression of the response away from the center of the object tends to insure that the largest peak in the correlation plane lies at or near the center of the object. With the response away from the center of the false class 2 objects suppressed as noted, their correlation response is generally low over the full correlation plane. Many additional variations to this basic algorithm exist. The most obvious one is to require that a zero response is obtained at $2d_s$ pixels away from the central correlation peak in the horizontal and vertical directions, respectively. This extension is generally not needed in the test cases we have investigated thus far. The choice of the shift d_s is made from analysis of the width of the autocorrelation function of the training set imagery.

Thus, to synthesize the filter h to satisfy Eqs. (4), we write h as a linear combination of all training set imagery

$$h(x,y) = \Sigma a_{f_i} f_i(x,y) + \Sigma a_{f_i} f'_i(x,y) + \Sigma a_{g_i} g_i(x,y) + \Sigma a_{g_i} g'_i(x,y) \tag{5}$$

We denote the full set of N_T training imagery by $\{z\}$ and the individual correlation images by $\mathbf{z}_n$. The filter h required is then defined by the coefficient vector $\mathbf{a}$ that solves

$$\mathbf{V}\mathbf{a} = \mathbf{u} = [\underbrace{1 \cdots 1}_{N_1}, \underbrace{0 \cdots 0}_{N_T - N_1}]^T, \tag{6}$$

where $\mathbf{V}$ is the N_T dimensionality VIP matrix for $\mathbf{z}$, and $\mathbf{u}_2$ contains N_1 ones to satisfy Eq. (4a) and $N_T - N_1$ zeroes to satisfy Eq. (4b)–(4d). We refer to this solution in Eq. (6) as SDF-1 [An exact correlation SDF, since it exactly satisfies Eq. (4)].

B. Dimensionality Reduction

The solution for h in Eq. (6) requires the solution of N_T linear algebraic equations for the N_T unknowns a_n. Since $\{f\}$ and $\{g\}$ are generally composed of different views or different distorted versions of one object or different types of object in one class and since such objects are not necessarily independent, the rank of $\mathbf{V}$ in Eq. (6) may not be full, the dimensionality of the problem can and must be included to solve the resultant linear algebraic equation in Eq. (3b) or (6). In addition, as various constraints (such as rotation, scale, aspect) are included, more training set imagery must be included, and thus N_T can increase significantly and so does the condition number (the maximum-to-minimum eigenvalue ratio) for the matrix $\mathbf{V}$. This makes the solution of Eq. (6) for $\mathbf{a}$ difficult and thus requires advanced linear algebraic solution methods. One can reduce the size N_T of the training set to alleviate this problem. However, if we reduce N_1 or N_2, the image representation degrades. If we reduce N_s, the shape of the correlation function degrades. Necessary solution methods to the general $\mathbf{Ae} = \mathbf{p}$ matrix-vector problem for the vector $\mathbf{e}$ that determines the weighting coefficients of the training set data for selected exogenous vectors $\mathbf{p}$ (analogous to the $\mathbf{u}_n$ vectors in prior equations) will be discussed. Our approach involves a reduction in the dimensionality of the problem with minimal loss in information and an approximate solution of the resultant linear algebraic equation (LAE) $\mathbf{Ae} = \mathbf{p}$.

To reduce the dimensionality of the matrix and hence the problem to be solved, we can represent the entire training set in a reduced dimensionality orthogonal basis function hyperspace. This can be achieved by computing the eigenvectors of the full training set. (These form our orthogonal basis function set.) We can then represent each image in this new space. However, the size of this space will generally be the size N_T of the training set, and hence the associated matrix size will not be reduced. Furthermore, computing the associated eigenvectors of this matrix is more complicated than solving the associated LAE. However, the optimum method (mean square error) to compress data into reduced dimensionality is by Karhunen-Loeve techniques on the eigenvectors.[8] If we compute the eigenvectors of the entire data set, we can retain only the most dominant ones (those with the largest eigenvalues) and use these as a reduced dimensionality space in which to represent all the training set images. However, these calculations for the full N^2-dimensionality correlation matrix (N^2 is the number of image pixels, whereas the rank of the associated correlation matrix is N_T) or for the VIP matrix of dimensionality N_T are generally more difficult than the solution of the original LAE. We thus simplify calculation of the eigenvectors by operating on subsets of the data as we now detail.

The eigenvectors are conventionally calculated from the correlation matrix, which is of size N^2. It is computationally much more efficient[9] to compute the eigenvectors from the VIP matrix of size $N_T \ll N^2$. These eigenvectors are of dimensionality N_T and are expressed in terms of the training set images. The eigenimages themselves (with dimensionality N^2) are easily obtained as a linear combination of the eigenvectors.

To avoid calculation of the eigenvectors of even the N_T dimensionality VIP matrix and insure better representation of the different object types, we consider the eigenvectors of only the unshifted images in each class. For class 1, we form the N_1 rank VIP matrix $\mathbf{V}_1$ [N_1 is typically 6 and thus much less than $N_T = N_s(N_1 + N_2)$, which is typically $5(6 + 6) = 60$]. We compute this $\mathbf{V}_1$ matrix for unshifted class 1 objects and then compute its eigenvectors. We then form the N_2 dimensionality VIP matrix $\mathbf{V}_2$ for class 2 objects (with $N_1 = N_2 = 6$; this matrix is also quite small, 6×6). For each of these VIP matrices, there are six eigenvectors. We retain the dominant three eigenvectors of each matrix, since these typically contain 93% of the information. (More eigenvectors can be retained if needed, and this can be determined from the eigenvalues.) Our tests show that in general the two or three (k value) dominant eigenvectors per VIP matrix are generally adequate. Another attractive feature of these two subset VIP matrices (one per class for the unshifted images in each class) is that the VIP matrices for shifted image subsets (i.e., all images in class 1 with a shift d_s) are the same as those for the original VIP matrix for the unshifted images. Hence the corresponding eigenvectors and eigenvalues of their VIP matrix are the same, as are the associated weighting coefficients used to obtain the eigenimages. Finally, the eigenimages are simply shifted versions of the eigenimages of the unshifted subset.

The steps in our dimensionality reduction procedure thus follow directly:

Step 1: For the $N_1 \times N_1$ and $N_2 \times N_2$ VIP matrices $\mathbf{V}_1$ and $\mathbf{V}_2$ for the unshifted class 1 and class 2 images compute their eigenvectors. We denote these eigenvectors by $\{\mu\}$ and $\{\mathbf{v}\}$ for each subset, respectively. These eigenvectors are of dimensionality N_1 and N_2, respectively. We retain the dominant k eigenvectors from $\mathbf{V}_1$ and $\mathbf{V}_2$. Typically, we use $k = 3$.

Step 2: Synthesize the eigenimages $\{\chi\}$ and $\{K\}$. Each of these eigenimages is of dimensionality N^2, and they are linear combinations (with coefficients that are the elements of $\{\mu\}$ and $\{\mathbf{v}\}$) of the training set images $\{\mathbf{f}\}$ and $\{\mathbf{g}\}$, respectively.

Step 3: Generate the eigenimages $\{\chi'\}$ and $\{K'\}$ for the shifted image subsets. There are four shifted subsets per object class for our case. This yields a total of $d = 10$ subclasses and k eigenimages per subclass. We denote these eigenimages by set $\{\phi\}$.

Step 4: Convert this set $\{\phi\}$ of $10k$ eigenimages into an orthogonal set $\{\psi\}$ by Gram-Schmidt[7] or other[10] techniques.

C. Least-Squares SDF-2 Synthesis

This procedure yields an orthonormal basis function set of $d' = 10k = 30$ (with $k = 3$) eigenimages. This is considerably reduced from N_T. Next, we project all N_T images onto this $\{\psi\}$ space and describe each training set image as a d' dimensionality vector in terms of these new $\{\psi\}$ basis functions. We hereafter denote the entire N_T set of training images by the vectors $\mathbf{z}_i$, and we refer to the images f_i, g_i, f_i', g_i', and the desired h discriminant function by their decomposition vectors **a, b, c, d,** and **e,** respectively, in $\{\psi\}$ space, i.e.,

$$\begin{aligned} &f_i = \Sigma a_{n_i}\psi_n, g_i = \Sigma b_{n_i}\psi_n, f_i' = \Sigma c_{n_i}\psi_n, \\ &g_i' = \Sigma d_{n_i}\psi_n, \text{ and } h = \Sigma e_n\psi_n. \end{aligned} \tag{7}$$

The desired filter h, denoted by **e,** is thus given by the solution of

$$\mathbf{Ae} = \mathbf{p} = [\underbrace{1 \cdots 1}_{N_1}, \underbrace{0 \cdots 0}_{N_T - N_1}]^T, \tag{8}$$

where **A** is $N_T \times d'$ with each row being one of the N_T training images in the d' dimensionality $\{\psi\}$ space. For our case, **A** is 60×30 (assuming six training set images per class, $N_S = 5$ and two classes), **p** is a N_T dimensionality vector that specifies the constraints in Eq. (4), and the desired **e** is a d' dimensionality vector. This leaves us with N_T constraints or equations to be satisfied by **e** (which defines h), which has only d' variables.

With $N_T > d'$, the matrix $\mathbf{A}^T\mathbf{A}$ is $d' \times d'$ and of rank d' and thus invertible. Thus we can apply a least-squares solution for this overdetermined problem. In this reduced dimensionality space $\{\psi\}$, we can now describe synthesis of SDF-2 (our least-squares approximation to SDF-1). The filter h (described by **e**) in this case minimizes

$$J(\mathbf{e}) = \sum_i (\mathbf{z}_i^T\mathbf{e} - p_i)^2, \tag{9}$$

where the summation is over N_T and where the projection value $p_i = 1$ for $\{f\}$ and is zero otherwise. Setting $\partial J/\partial \mathbf{e} = \mathbf{O}$, the filter solution is found to be

$$\mathbf{e} = (N_T)^{-1}\mathbf{R}^{-1}\sum_i b_i\mathbf{z}_i, \tag{10}$$

where **R** is the $d' \times d'$ (30×30) correlation matrix for the full $\{\mathbf{z}\}$ training set,

$$\mathbf{R} = (N_T)^{-1}\sum_i \mathbf{z}_i\mathbf{z}_i^T. \tag{11}$$

This SDF-2 is thus a least-squares approximation to the exact SDF-1, since its projection values will be as close to 1 and 0 as possible.

D. Peak-to-Sidelobe Ratio Maximized Correlation SDF-3 Synthesis

A considerably different correlation SDF that maximizes the correlation plane SNR can be synthesized by viewing the projection of h on class 1 data $\{f\}$ as a signal and the projections on $\{g\}$, $\{f'\}$, and $\{g'\}$ data as noise. We refer to this as the peak-to-sidelobe ratio (PSR) maximized correlation SDF-3 or simply SDF-3. The mean square value of $\langle \mathbf{f} \cdot \mathbf{h} \rangle$ for all images in $\{f\}$ is computed as

$$E\{(\mathbf{e}^T \cdot \mathbf{a})^2\} = \mathbf{e}^T\mathbf{R}_f\mathbf{e}, \tag{12}$$

where

$$\mathbf{R}_f = (N_1)^{-1}\sum_i \mathbf{a}_i\mathbf{a}_i^T \tag{13}$$

is the correlation matrix of the images $\{f\}$, and summation in Eq. (13) is over the N_1 training set images in $\{f\}$. The mean square value of the projections onto $\{g\}$, $\{f'\}$, and $\{g'\}$ are similarly found, and their sum is

$$E\{(\mathbf{e}^T \cdot \mathbf{b})^2\} + E\{(\mathbf{e}^T \cdot \mathbf{c})^2\} + E\{(\mathbf{e}^T \cdot \mathbf{d})^2\} = \mathbf{e}^T\mathbf{R}_n\mathbf{e}, \tag{14}$$

where $\mathbf{R}_n$ is the sum of the correlation matrices for each of these three image sets, which we view as noise, i.e.,

$$\mathbf{R}_n = (N_2)^{-1}\sum_{i=1}^{N_2} \mathbf{b}_n \cdot \mathbf{b}_i^T + (4N_1)^{-1}\sum_{i=1}^{4N_1} \mathbf{c}_i \cdot \mathbf{c}_i^T + (4N_2)^{-1}\sum_{i=1}^{4N_2} \mathbf{d}_i \cdot \mathbf{d}_i^T. \tag{15}$$

In Eqs. (13) and (15), each **R** is of size $d' \times d'$ ($d' = 30$ in our example).

There are actually nine separate noise terms, those due to $\{g\}$ and those due to each of the four shifted $\{f'\}$ and four shifted $\{g'\}$ image subsets. Each of these noise terms has a mean square projection with h of the form of Eq. (12). We desire to maximize the PSR of Eq. (12) with respect to each of these nine terms. The simultaneous maximization of these nine PSRs is not possible. Thus we consider maximizing the peak (12) with respect to the sum of the nine noise terms as in Eq. (14). This does not maximize each PSR, but it results in a problem that one can solve.

With the above preliminaries, the SNR or PSR to be maximized is the ratio of the signal or peak correlation value in Eq. (12) to the noise or sidelobe values in Eq. (14):

$$\text{PSR} = \frac{\text{mean square signal or correct peak value}}{\text{sum of the mean square false peak and sidelobe values}}. \tag{16}$$

From Eqs. (12) and (14), this is equivalent to the maximization of

$$\text{PSR} = \frac{\mathbf{e}^T\mathbf{R}_f\mathbf{e}}{\mathbf{e}^T\mathbf{R}_n\mathbf{e}}. \tag{17}$$

The solution for **e** that maximizes Eq. (17) can be obtained by use of Lagrange multipliers and is well known to be the dominant eigenvector solution to the generalized eigenvalue problem:

$$\mathbf{R}_f\mathbf{e} = \lambda\mathbf{R}_n\mathbf{e}. \tag{18}$$

Our SDF-3 synthesis thus operates on the same re-

duced eigenimage hyperspace $\{\psi\}$ of dimensionality $d' = 30$ (for our example) as used in SDF-2 synthesis but with the different SDF-3 maximization criteria of Eq. (17).

This SDF-3 is related to other modified MSFs and statistical correlation filters. In earlier work,[11] the optimal filter for a two-class problem (with the second class modeled as white noise) was considered. The mean square central correlation peak for class one images with respect to white noise was maximized (i.e., subject to a constant filter energy). The solution was the principal component of the class one images. Later work[12] extended this concept to filter synthesis by maximizing the ratio of the mean square central correlation peak for one class to the sum of the mean square central peaks for the second class and white noise. In the system designed,[12] the eigenimages of class 1 and class 2 were used, but no composite basis set was employed. Our work includes a completely orthogonal basis function set as well as use of shifted imagery with attention to correlation peak shape.

IV. Initial Results and New Classification Parameter Selection Criteria

Before discussing several details of correlation SDFs, we show typical output correlation plane data obtained. This is intended to provide motivation for the parameters used.

A. Representative Output Correlation Planes for Correlation SDFs

In Fig. 1, we show representative true and false class correlation patterns obtained with projection SDFs. The true class 1 correlation patterns generally consisted of a broad peak above threshold and with false class 2 object correlations having various peaks of large value above threshold displaced from the origin. In Figs. 2 and 3, we show the correlation SDF-1 and correlation SDF-3 outputs, respectively, for the same two test images with the filters synthesized from the same training set images (but using our new correlation SDF algorithms). As Fig. 2(a) shows, the correlation response for a true class 1 object has a very dominant and sharp correlation peak with zero values d_s pixels away (as specified by the algorithm) and peaks of negligible height elsewhere in the correlation plane. Comparing Fig. 2(a) with 1(a), the correlation peak obtained with SDF-1 is much sharper and better defined. The response in Fig. 3(a) for SDF-3 is even better (as expected, since this SDF algorithm maximizes the correlation plane PSR). We thus expect better true class 1 target detection with correlation SDFs. In SDF-1, the correct peak is 0.75, and the largest other peak is 0.36 in height. In SDF-3, the central peak is much larger (3.62) than with SDF-1, and the largest peak anywhere in the correlation plane (3.68) is only one pixel away from the central value.

Figures 2(b) and 3(b) show data for these correlation SDFs vs the same false class test input used in obtaining Fig. 1(b). In both figures, we see that the entire correlation plane has no significant peak, and we find that the center of the plane and four points d_s pixels away have especially low values (as specified by both correlation plane SDF algorithms). The largest peak value anywhere in Fig. 2(b) is 0.49 and considerably below the true class peak value. In Fig. 3(b), the

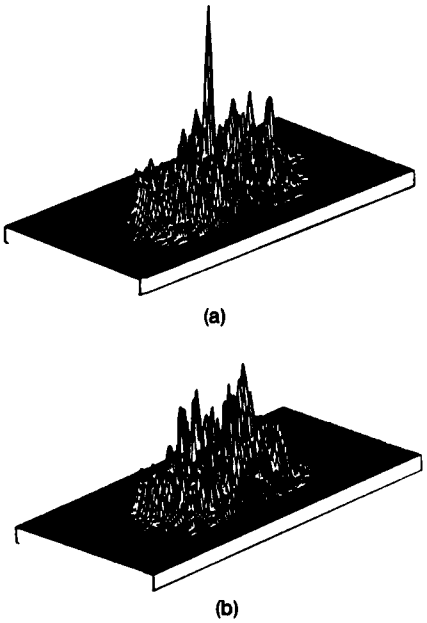


Fig. 2. True (a) and false (b) class correlation peak outputs for the exact correlation SDF-1 filter (using the same training set and test set as in Fig. 1).

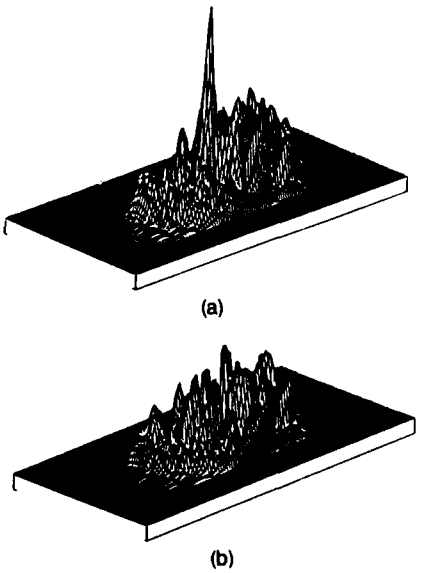


Fig. 3. True (a) and false (b) class correlation peak outputs for the PSR-maximized SDF-3 (using the same training set and test set images as in Fig. 1).

largest peak value anywhere is 1.89 and is again considerably below the 3.62 true peak value. Thus, in both cases, true class peaks are much more easily detected, and the full correlation plane response for false targets is greatly suppressed. Hence we expect correlation SDFs to provide much fewer false alarms for class 2 objects, and we expect SDF-3 to yield the most easily detected output peaks. SDF-1 still has distinct advantages (in terms of its extension to multilevel filters) because its output peak value can be specified. Conversely, the peak value for SDF-3 is not specified by the algorithm. (Thus this filter is not easily extended to multilevel concepts.)

B. Correlation Plane Classification Parameters

The entire correlation SDF formulation can also be described in terms of producing more meaningful and useful output classification parameters and statistics. In the projection SDF, a simple correlation plane threshold T_T was the parameter used to determine if a true correlation (class 1 object) was present. In a correlation SDF, the output correlation plane requirements of zero (for SDF-1 and SDF-2) or low (for SDF-3) correlation plane values d_s pixels from the central peak were used to suppress sidelobe peaks (especially with false class 2 objects). However, these design rules can equally be viewed as placing requirements on the shape of a true correlation function.

With this viewpoint in mind, we can then consider measuring the PSR = C for candidate regions of the correlation plane (where a peak above threshold or a large peak value) occurs. We can then compare this C measure to several C_T thresholds to determine if a candidate correlation peak is a true correlation peak or a noise spike. The two C measures we consider are

$$C_1 = \frac{\text{peak value}}{\text{average value in a window (excluding the peak)}}, \tag{19}$$

$$C_2 = \frac{\text{average value of the peak and its four neighbors}}{\text{average value in a window (excluding peak and its neighbors)}}. \tag{20}$$

The window in Eqs. (19) and (20) is the $(d_s + 1) \times (d_s + 1)$ region about the peak. The PSR measure C_1 is the classic PSR defined with a specific window region denoted by the algorithm. The measure C_2 accounts for the observed fact that many noise points in the correlation plane are a single isolated spike, whereas true target correlations have significant values adjacent to the peak point (hence the modified form of the numerator in C_2). The combination of a T_T peak threshold and a PSR threshold C_T should thus provide improved confidence in selection of true correlation peaks. These T_T and C_T thresholds can be determined from training set image tests and the noise expected in a given application. In applications with more variation in the modulation and intensity level of the data, adaptive T_A thresholds are needed. These thresholds are functions of the mean m and variance σ of the correlation plane pattern obtained. We discuss these thresholds and our specific classification correlation plane parameter selection in the following subsection.

C. Correlation Classification Parameter Selection

For the fixed threshold T_T, we form the SDF and test it on the training set images in each class and measure the means m_{T1} and m_{T2} of the highest correlation peaks for the class 1 and class 2 training set images. The peak threshold is then set at a linear combination of these mean peak values as

$$T_T = \alpha_T m_T + B_T m_{T2}, \tag{21a}$$

where

$$\alpha_T + \beta_T = 1. \tag{21b}$$

By adjusting α_T and β_T, we can detect more true class targets or reject more false class targets (as a specific scenario requires). This means m_{C1} and m_{C2} for the C ratio for the class 1 and class 2 training set images are also measured, and the C threshold (C_{T1} or C_{T2}) is set at a linear combination of these means as

$$C_T = \alpha_C m_{C1} + \beta_C m_{C2}, \tag{22a}$$

where

$$\alpha_C + \beta_C = 1. \tag{22b}$$

The choices for the α and β coefficients in Eqs. (22) again depend on the performance desired.

Nontraining set images have (as expected) lower peak values and C values than those obtained from training set images. Thus in general we select $\alpha < \beta$ in all thresholds. This insures that nontraining set true class 1 images will be detected. The variation of the peak and C data, the number of training set images used, and any overlap of the training set data also affect the α and β choices. In general, we use $\alpha \geq 0.25$ to reduce the number of false targets detected and the number of points for which C tests must be performed. In cases where high noise is expected, α is adjusted accordingly. If the training set is large, our confidence in the mean estimates for m_{C1} and m_{C2} improves, and α can be increased with high confidence.

In many cases, the intensity and modulation level of the image may vary significantly in testing and among the training set. In this case, a reliable fixed T_T threshold may not exist. (The variation of m_{T1} and m_{T2} and overlap of the two classes of data will denote this case.) When this occurs or in situations where high noise levels are expected, the mean m and variance σ of the entire correlation plane are computed for each training set image and an adaptive threshold

$$T_A = m + N\sigma \tag{23}$$

is used, where N is chosen from training set data. During tests, m and σ for each full output correlation plane for each test image are computed, and the threshold level in Eq. (23) is set adaptively at N standard deviations above the measured mean for the given test image and its variance σ.

The C ratio is not affected by variations in the intensity and modulation of the data. However, spatially

varying modulation (as can occur with IR imagery) can cause variations in the shape of the correlation peak and hence in the C values. An adaptive C_A threshold can also be considered, but we have not found this necessary.

The SDF-3 algorithm does not specify a peak value, and thus a C test is generally best for this SDF. Since we expect this C ratio to be much better than that for SDF-1 and SDF-2 (since SDF-3 optimizes PSR), this measure is generally quite successful. However, since a large PSR generally relates to a large peak value, the simpler T_T or T_A threshold test is often adequate for use with SDF-3 also.

The 3-D distortions between the different images can only be described statistically, and this is only possible if a probability distribution can be assumed and modeled. This is generally not possible, and thus we must select our thresholds from training set data by the methods noted above.

V. Summary and Conclusion

The synthesis of a modified matched spatial filter for a shift-invariant multiobject correlator has been described. This is an extension of projection synthetic discriminant functions that provide control over the shape of the output correlation function. Synthesis of three forms of these correlation SDFs was detailed. These include an exact SDF-1 (that specifies exact values for the correlation peak and several points around it), a least-squares SDF-2 (that approximates SDF-1 and that is useful when the exact SDF-1 solution is difficult to obtain), and a new SDF-3 that optimizes the peak-to-sidelobe ratio of the output correlation. (Detection of such an output correlation peak is significantly simplified.) With control of the shape of the output correlation peak for true and false class objects provided by these new correlation SDFs, new classification measures considering the mean, variance, and shape of the output correlation pattern are advanced to allow true correlation peaks to be discriminated from erroneous noise spikes. Initial demonstrations of these new filters show the significantly better correlation patterns that result.

References

1. A. VanderLugt, "Spatial Detection by Complex Spatial Filtering," IEEE Trans. Inf. Theory **IT-10**, 139 (1964).
2. D. Casasent, "Coherent Optical Pattern Recognition: A Review," Opt. Eng. **24**, 26 (Jan. 1985).
3. B. V. K. Vijaya Kumar and C. W. Carroll, "Loss of Optimality in Cross Correlators," J. Opt. Soc. Am. **1**, 392 (1984).
4. D. Casasent and D. Psaltis, "New Optical Transforms for Pattern Recognition," Proc. IEEE **65**, 77 (Jan. 1977).
5. J. R. Leger and S. H. Lee, "Hybrid Optical Processor for Pattern Recognition and Classification using a Generalized Set of Pattern Functions," Appl. Opt. **21**, 274 (1982).
6. D. Casasent, "Unified Synthetic Discriminant Function Computational Formulation," Appl. Opt. **23**, 1620 (1984).
7. R. Duda and P. Hart, *Pattern Classification and Scene Analysis* (Wiley, New York, 1973).
8. K. Fukunaga and W. Koontz, "Application of the Karhunen-Loeve Expansion to Feature Selection and Ordering," IEEE Trans. Comput. **C-19**, 311 (1970).
9. H. Murakami and B. V. K. Vijaya Kumar, "Efficient Calculation of Primary Images from a Set of Images," IEEE Trans. Pattern. Anal. Machine Intell. **PAMI-4**, 511 (1982).
10. H. J. Caulfield and R. Haimes, "Generalized Matched Filtering," Appl. Opt. **19**, 181 (1980).
11. B. V. K. Vijaya Kumar, D. Casasent, and H. Murakami, "Principal Component Imagery for Statistical Pattern Recognition Correlators," Opt. Eng. **21**, 43 (Jan./Feb. 1982).
12. D. Casasent, B. V. K. Vijaya Kumar, and H. Murakami, "A Correlator for Optimum Two-Class Discrimination," Electro-opt. Syst. Des., **248**, 321 (1981).

The support of this research by Independent Research and Development Funds of General Dynamics-Pomona Division and the Air Force Office of Scientific Research (grant AFOSR-84-0293) is gratefully acknowledged, as are the Westinghouse Corp. furnished equipment at the CMU Center for Excellence in Optical Data Processing used in this work.

CHAPTER 4:

"OPTICAL ARTIFICIAL INTELLIGENCE PROCESSORS"

Proc SPIE
Vol. 700
IOCC '86, Israel

"OPTICAL ARTIFICIAL INTELLIGENCE PROCESSORS"

David Casasent

Carnegie Mellon University
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213 (USA)

ABSTRACT

We briefly highlight recent CMU (Carnegie Mellon University) research on optical Artificial Intelligence (AI) processors for scene analysis. This work includes new shift-invariant AI correlators, plus symbolic, model-based, associative memory, knowledge-based, relational-graph, and neural optical processor results.

1. INTRODUCTION

Optical AI has received major attention during the past year. In Sections 2-7, we review recent CMU research on seven aspects of this topic.

2. KNOWLEDGE ORGANIZATIONAL BASE [1]

Advanced computers must use declarative and rule-based knowledge. We have distinguished between explicit and implicit declarative (facts) and procedural knowledge (rules) and their use and role in optical symbolic and other processors. We have also emphasized the importance of and the preference for shift-invariant processors and pattern recognition (rather than logical) applications. Figure 1 shows the general diagram of the system which distinguishes pattern recognition from logical symbolic processors. This distinction, plus the need for shift-invariance has not appeared to be fully appreciated within the optical AI community. The separate blocks in Figure 1 note the role of an optical pattern recognition processor and an optical logical processor within a symbolic optical data processor environment. A projection or correlation processor is considered with different symbol inputs as filters to the pattern recognition system. The outputs are then operated upon by a symbolic logic processor to derive new filter inputs or to invoke final decisions.

3. RELATIONAL GRAPH AND DECISION NET OPTICAL PROCESSING [2]

We have recently advanced the concept and use of a high-dimensionality optically-generated feature space for pattern recognition. Figure 2 shows this system. At P_1 the input object is fed to the system. Its Fourier transform magnitude is produced at P_2, where a Coordinate Transformation (CT) mask is also provided. The CT of the input image is provided at P_3. The magnitude Fourier transform of P_3 is produced at P_4. At this plane we also incorporate Linear Discriminant Function (LDF) filters. The specific CT performed is a polar-log CT on the input P_2 data. We note that the optical feature space produced at P_4 is performed and implemented in parallel. The Computer Generated Hologram (CGH) included at P_4 forms several linear combinations of these features, from which the object class identification is possible. The P_4 to P_5 system is the relational graph processor. Attractive features of this system are its full parallelism, the ability to use different features at different nodes of the relational graph and the significant reduced number of output detectors possible (especially with multi-level filter encoding). Associated with this effort, we have devised automated techniques to arrange a fact-based database into a rule-based

one and to automate selection of the proper decisions to be made at each node and the features to be employed at each node of our relational graph. We have also distinguished between graph designs based upon structural features, rather than functional attributes of the objects. Both multi-decision and binary relational graph designs have been achieved and tested with excellent initial results for the specific case of a polar-log CT of the magnitude Fourier transform feature space input. Another type of decision net processor that operates on the output projections from the pattern recognition unit in Figure 1 has also been devised (Figure 3) [1].

4. MODEL-BASED PROCESSORS [3]

An efficient representation of a 3-D object as a set of polygon vertices in an object-centered coordinate system has been developed. From this, we can produce on-line binary and range images of the object at any 3-D orientation and scale. This allows on-line smart filter synthesis and the use of reference imagery, which is much preferable to processing lists of lines and vertices in an input image (since it allows real time processing on realistic input imagery). This model-based optical processor also provides very efficient image storage with only one reference object per object class required. The use of this technique in model-based associative and symbolic processors is discussed later and has been detailed elsewhere [3].

5. OPTICAL NEURAL PROCESSORS [4]

We have recently studied the optical Hopfield neural processor and found many deficiencies associated with one version of this matrix system. These include: only one stable state for general vector inputs, a non-binary matrix required, the need for iterative processing of this matrix, the presence of many false memory states, and the major fact that this processor does not necessarily converge to the nearest neighbor. Alternate Hopfield memory matrices and the use of input vectors with few ones appear to perform adequately. We have devised a direct storage nearest neighbor associative processor. The matrix for this case is binary by design, the matrix size is significantly reduced, it can accommodate many stable states, it is a true nearest neighbor processor, it provides the Hamming distance and such processors can be cascaded. This memory matrix is used in some of our associative, symbolic, and decision net optical system work.

6. ASSOCIATIVE PROCESSORS [3,5]

Our associative memory work has involved distortion-invariant multi-class pattern recognition. We have devised new and efficient techniques to employ associative processors to achieve these goals. These include new memory synthesis methods, the use of dual auto and hetero associative processors, a vector inner product associative processor (Figure 4) and efficient matrix update and synthesis techniques, plus quantitative data on the use of each system for distortion-invariant pattern recognition. In the system of Figure 4, the input vector u' at P_1 is provided. The key recollection vectors $\underline{u}_k$ are encoded on separate channels at P_2 and the recollection vectors $\underline{v}_k$ are similarly encoded on separate channels at P_4. The scalar outputs at P_3 are the projection of the input vector onto the various key vectors at P_2. These scalar P_3 outputs can be thresholded if desired. These scalar P_3 outputs are then multiplied by the recollection vectors $\underline{v}_k$ encoded on separate channels at P_4. The results of these P_4 projections are then summed along vertical elements onto detectors at P_5. The P_5 position(s) with peak values denote the encoded input object class. Both auto and hetero associative versions of this system are possible as well as the use of nonlinear operations at P_3 and P_5. The incorporation of such nonlinear operations can significantly enhance the versatility of the system and improve its performance. This vector inner product architecture allows much easier updating of the processor than is possible in the conventional vector outer product formulation of such associative memory matrices.

7. SYMBOLIC CORRELATION PROCESSORS [1,6]

A hierarchical optical symbolic processor using correlation, smart filters, encoded outputs, and our model-based knowledge base is shown in Figure 5. This figure employs a smart activity-filter at P_2 of Figure 5. The correlation output of this smart "Peak-to-Sidelobe Ratio" (PSR) filter or activity filter at P_2 denotes where regions of activity

exist in the input P_1 scene. These peak locations denote which points the P_5 multiple correlation plane output analysis system should investigate. The multiple frequency correlation planes at P_5 employing smart filters at P_4 can also be encoded to designate L output correlation levels with F output filters. This provides the ability to recognize a large L^F number of input object classes. Both correlation and projection filters are employed at P_4 to achieve this P_5 output. With only F = 4 filters and L = 10 levels, a 10,000 class problem can potentially be solved on such a processor, with high probability of detection and low error rates. The orientation of the objects in the scene as well as their class can also be obtained from this system output.

This research has been supported by grants from the Defense Advanced Research Project Agency and the Air Force Office of Scientific Research. We gratefully acknowledge their support of this research.

REFERENCES

1. D. Casasent and E. Botha, "Knowledge in Optical Symbolic Pattern Recognition Processors", **Optical Engineering**, *Special Issue*, accepted (publication January 1987).

2. D. Casasent and A.J. Lee, "An Optical Relational-Graph Rule-Based Processor for Structural-Attribute Knowledge Bases", **Applied Optics**, *Special Issue*, accepted (publication late 1986).

3. D. Casasent and S.A. Liebowitz, "Model-Based Knowledge-Based Optical Processors", **Applied Optics**, *Special Issue on Artificial Intelligence and Non-Numerical Processing*, Vol. 26, submitted (tentative publication January 1987).

4. B.L. Montgomery and B.V.K. Vijaya Kumar, "Nearest-Neighbor Non-Iterative Error Correcting Optical Associative Memory Processor", **Proc. Soc. Photo. Opt. Instr. Engrs.**, Vol. 638 (March-April 1986).

5. D. Casasent and B. Telfer, "Distortion-Invariant Associative Processors", **Proc. Soc. Photo Opt. Instr. Engrs.** (August 1986).

6. D. Casasent, "Optical AI Symbolic Correlators: Architecture and Filter Considerations", **Proc. Soc. Photo. Opt. Instr. Engrs.**, Vol. 625 (January 1986).

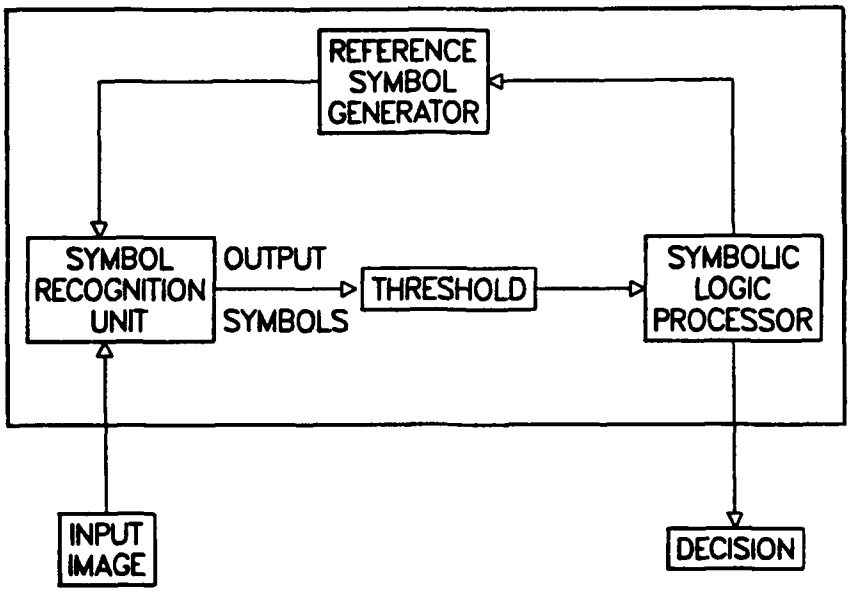


FIGURE 1: Optical symbolic inference processor using procedural knowledge

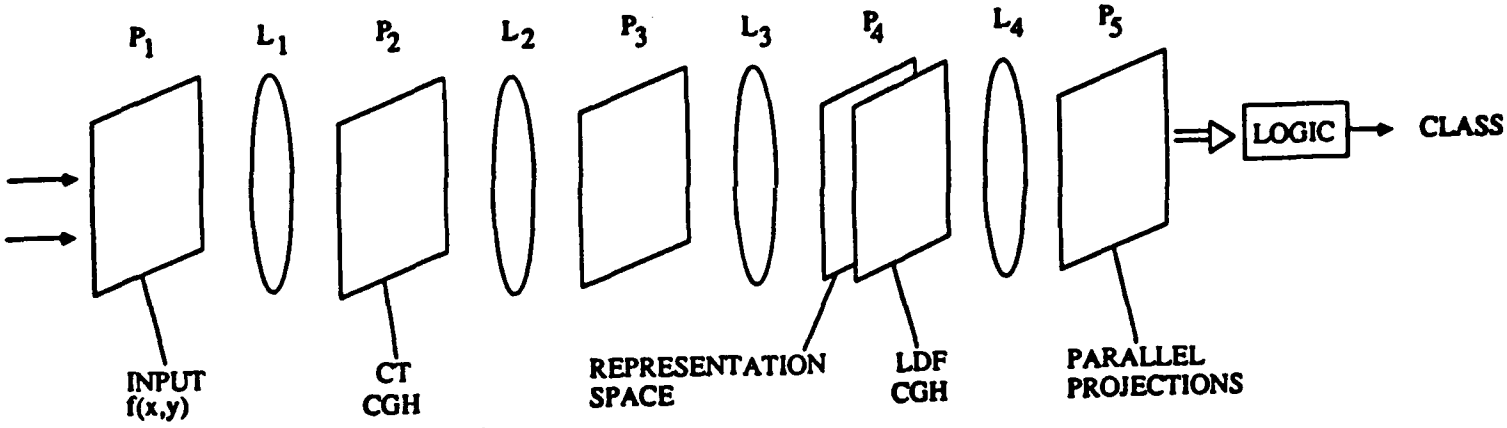


FIGURE 2: Optical feature space and relational graph computer generated hologram processor

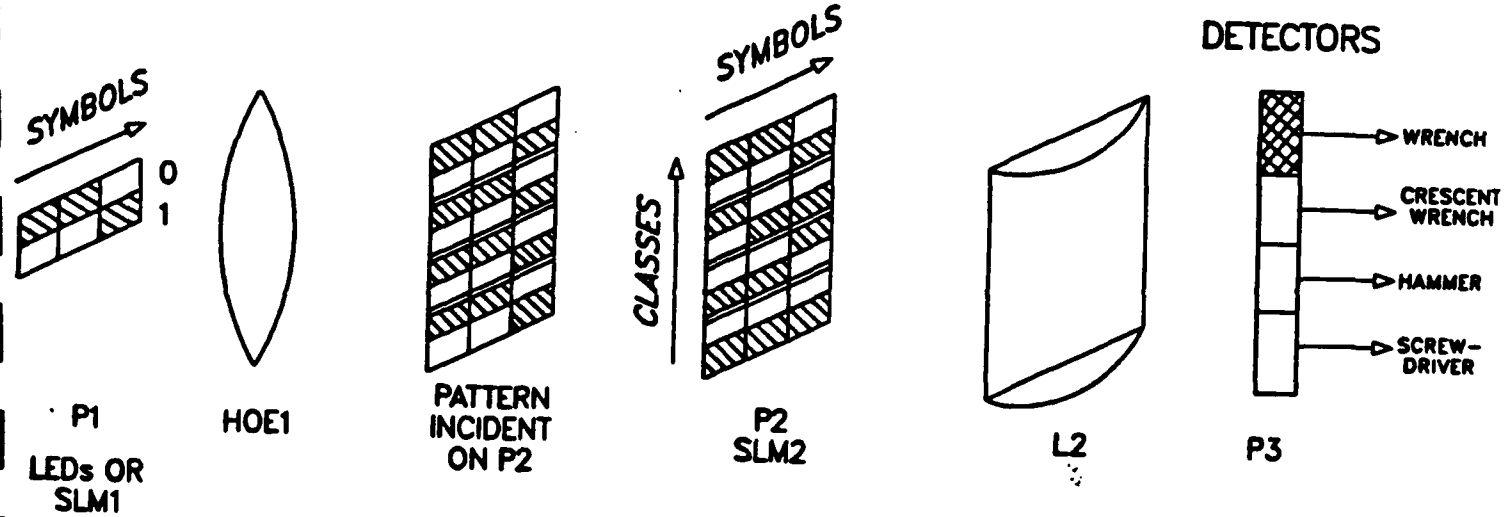


FIGURE 3: Optical decision net processor

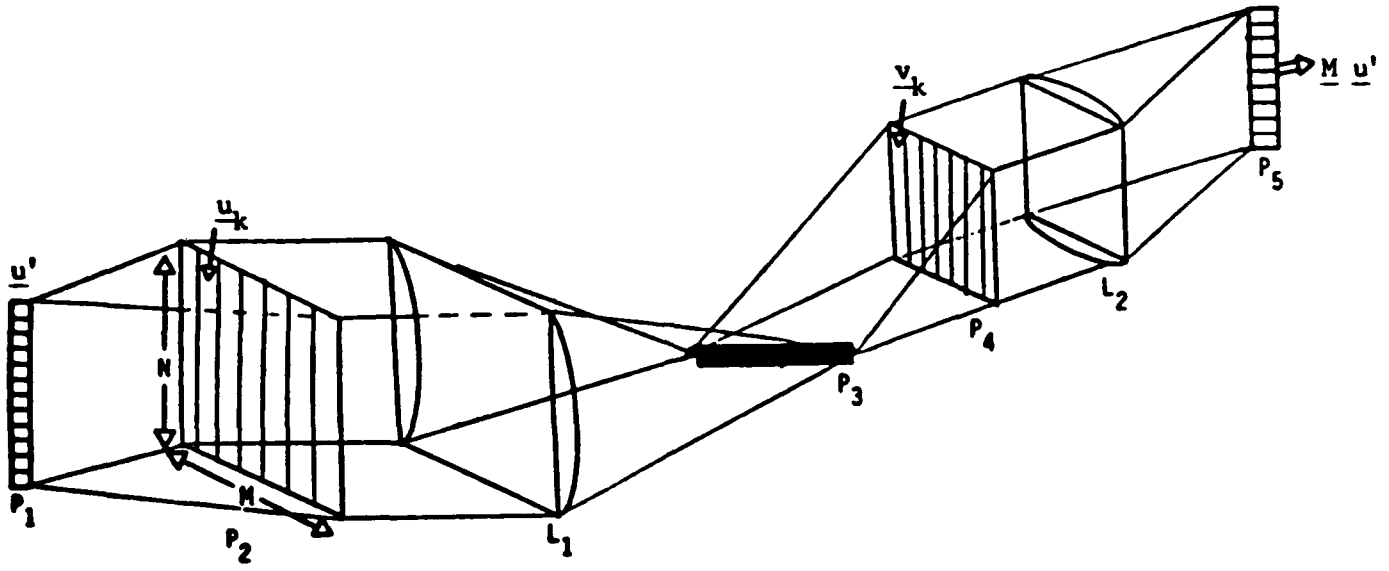


***FIGURE 4**: Vector inner product associative matrix multi-channel distortion-invariant processor*

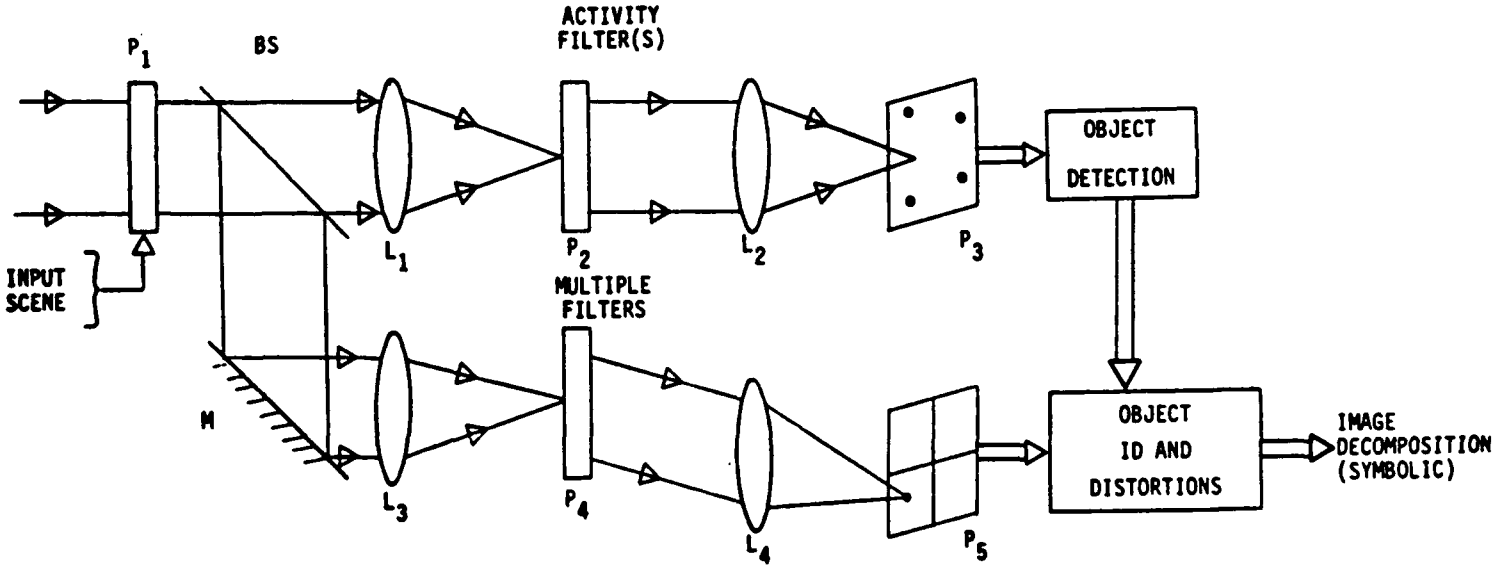


***FIGURE 5**: Multiple PSR (P_8) and projection (P_4) filter optical artificial intelligence object identification and distortion estimation processor*

CHAPTER 5:

"DIRECTED GRAPH FOR ADAPTIVE ORGANIZATION AND LEARNING OF A KNOWLEDGE BASE"

Reprinted from Applied Optics, Vol. 27 page 534, February 1, 1988

Directed graph for adaptive organization and learning of a knowledge base

David Casasent and Edward Baranoski

A directed graph is considered for organization of a knowledge base for neural, associative, model-based, and other advanced processors. Its ability to self-organize itself, delete old information, and add new information and its many interconnections make it most suitable for optical realization and use in advanced neural and adaptive optical processors. An alphanumeric image space example is used as a case study, and an optical processor architecture to achieve this with impressive performance is discussed.

I. Introduction

Considerable attention[1,2] has recently been given to knowledge bases, model bases, neural networks, and their optical realization. All these architectures and algorithms require and demand some method for organizing data. Toward this end, we advance the concept of a directed graph[3] with attention to its self-organizing and adaptive properties and its use of interconnections. Since it allows multiple processors and a large number of interconnections, it is most suitable for optical realization.[4] Section II introduces the directed graph and terminology. Section III extends these basic concepts to general data classification problems. Section IV discusses its adaptive properties, its use in knowledge base organization, and its suitability for optical realization by interconnection. Section V advances optical architectures that achieve the broadcasting and parallel multiprocessor properties suitable for realization of optical directed graphs. A directed graph case study (alphanumeric character recognition) is employed throughout and summarized and detailed in Sec. VI.

II. Terminology

The terminology widely used in graph theory is briefly summarized for completeness, and an introduction to a directed graph is provided. As background, we note that tree classifier structures are the most widely known hierarchical decision technique[5] and have many variations.[6,7] However, all hierarchical techniques have significant problems,[8,9] such as the need for backtracking when an error is made at a high-level node. The origin of this problem is the rigid structure of a tree classifier, its lack of interconnections, and the inability to reach all nodes from a given node without backtracking up the tree and then back down it. Adding new classes or data to a tree (adding new nodes) is cumbersome and *ad hoc*. A directed graph (Fig. 1) is quite different, since it exhibits many interconnections and each node is reachable from all other nodes. Table I lists typical terms used in describing directed graphs.

In the problem we consider, the nodes of the graph represent different classes in a pattern recognition problem, and the problem is to find the node or class of the input object under question. The input data to be processed by the graph are typically symbolic, although any object description is possible. In the case study we consider, the input data is an iconic image pixel description of the object to be classified. From Fig. 1, we immediately see that a directed graph has a large number of interconnections (arcs), and this alone makes it most attractive for an optical realization. Figure 1 also shows a disconnected graph (see Table I) to the right. Such graphs do present some problems, which can be handled, as we discuss later.

The adjacency of two nodes (see Table I) is an important concept as it describes the connectedness of a graph (which nodes can be reached from a given node). Figure 2 shows a typical adjacency matrix A_1. For a graph with N nodes, A_1 is $N \times N$. If an entry $a_{ij} = 1$, it indicates that an arc or path connects node i to node j. The ones per row indicate the nodes one can reach from the given node. The ones per column indicate which nodes can reach the node in question. A set of matrices $\{A_n\}$ is used to define those nodes reachable from

The authors are with Carnegie Mellon University, Department of Electrical & Computer Engineering, Center for Excellence in Optical Data Processing, Pittsburgh, Pennsylvania 15213.

Received 25 September 1987.
0003-6935/88/030534-07$02.00/0.

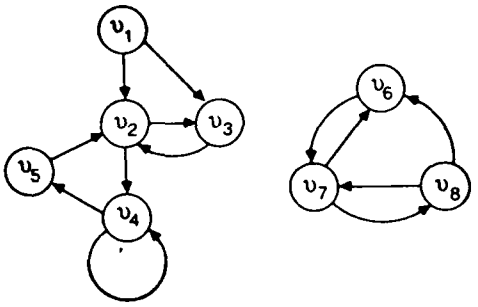


Fig. 1. Directed graph example.

$$A = \begin{bmatrix} 01100000 \\ 00110000 \\ 01000000 \\ 00011000 \\ 01000000 \\ 00000010 \\ 00000011 \\ 00000110 \end{bmatrix}$$

Fig. 2. Example of an adjacency matrix.

each node by a path of length (cardinality) n or less. Thus A_1 describes adjacent vertices or nodes and the in-degree and out-degree (see Table I) of each node (i.e., it describes the graph). A_2 denotes those nodes that one can reach with paths of length 2 or less etc. One can easily form A_{n+1} from A_n by

$$A_{n+1} = A_n \otimes (I \oplus A_1), \tag{1}$$

where $\otimes$ denotes binary matrix multiplication (with numerical multiplication and addition replaced by the logic AND and OR operations, respectively), and $\oplus$ denotes a matrix logic OR. For some m, a stable result $A_{m+1} = A_m$ occurs, and the resultant extent matrix $A_m = E$ contains the set of vertices reachable from any node by any path (i.e., it describes the connectedness of the graph).

III. Object Classification Example

To demonstrate the use and structure of a graph, we consider its use in classifying input data. The input data is a vector (an iconic image, feature vector, symbolic description, etc.), and each node also has a vector associated with it. The example we use is the set of sixty-two alphanumeric characters (twenty-six upper and lower case letters and ten numbers). Each character is a 5×9 dot matrix zero-filled and lexicographically ordered and zero-padded to form sixty-four element binary iconic vectors. In forming the graph, we want to map each class to a node, determine the outgoing and ingoing arcs to and from each node, and maintain the connectedness of the graph. This is done in an organized and automatic fashion. Adjacent vertices should indicate more similarity or connectedness between classes. In using the graph, we want to find the vertex (node) that best matches the input data vector. In more general terms, nodes represent knowledge, and arcs represent the similarity of knowledge.

Table I. Directed Graph Terminology

Nodes (v_n) = Objects or Classes
Arcs = Interconnections (Uni-directional)
Adjacency = Two Nodes Joined by and Arc
Indegree = Number of Arcs Entering Node
Outdegree = Number of Arcs Leaving Node
Loop = Arc from Node to Itself
Path = Route from One Node to Another Through Arcs
Cardinality = Number of Arcs in a Path
Circuit = Path with Same Starting and Ending Node
Disconnected Graph = Graph with Isolated Nodes Unreachable from Other Nodes

We first discuss the use of a directed graph in object classification. Here the graph has been constructed and exists. By a vector inner product (VIP) or correlation, we compare the input data to be classified with a number of starting nodes in the graph and select the best match. We then compare the input vector and data vector associated with all nodes adjacent to the best entry node. The node most similar to the input data determines the arc we move along and the next node we enter. We then compare the input vector to the vectors for all nodes adjacent to the new node. This procedure continues until some vertex is more similar to the input data than are any vertices adjacent to it. If this similarity is above threshold, we assign the input to the class represented by that node. If the threshold is not exceeded, this node is a local maximum. We then perturb the system and jump to a new region of the graph and continue the process. If every node has been examined and none exceeds our threshold, we either add a new node or restructure the graph. These operations are easily achieved as we detail in Sec. IV.

Searching through a directed graph is similar to traversing a tree classifier, except the larger number of interconnections in a graph allow much more flexibility in reaching one node from a given node. If the graph is fully connected, any vertex can be reached from any vertex. We modify the conventional graph by using *meta*-vertices (which do not represent a class) to connect subgraphs and determine the nodes at which to enter the graph. We implement the directed graph with an M-channel parallel optical processor, and thus we enter the graph initially at M nodes in parallel. For our alphanumeric character recognition problem, we use $M = 4$, and we use the four *meta*-vertices masks shown in Fig. 3 as the data vectors for the initial input entry nodes. These masks determine the number of input pixels in each quadrant. From this, we select one of four initial entry nodes to the

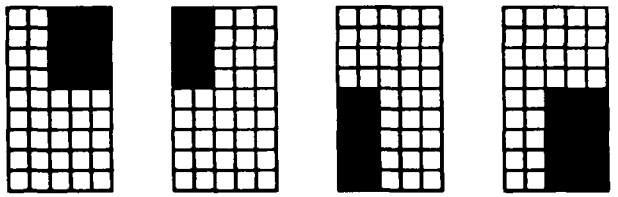


Fig. 3. Data vectors used for the $M = 4$ *meta*-vertices in our example. They correspond to counts of the number of pixels in each quadrant of the input data image vector.

graph. This extends the parallelism of the processor to the graph entry level also. As we shall quantify, this significantly reduces the path length to a solution and hence the search time required.

Several pitfalls that exist in a directed graph classifier are now noted, and the techniques we use to overcome them are addressed. Disconnected subgraphs can exist (we handle them with *meta*-vertices, the use of multiple starting nodes, and perturbation jumps). Several vertices may have an in-degree of zero (they are not reachable from any vertex); our graph synthesis technique ensures that such cases do not arise. Local maxima can occur in the maximum-ascent search performed. We avoid becoming trapped in local maxima by perturbation jumps rather than backtracking or simulated annealing[10] (which allows one to move to less optimal nodes with finite probability). We found that perturbation jumps and hard decisions were preferable in our study. Cyclic paths (circuits in Table I) can exist when a diagonal element of the extent matrix E is nonzero. This corresponds to the case when a node is reachable from itself. The maximum ascent algorithm cannot return to the same point, and thus such circuits will not be traversed in practice. However, they are redundant structures that can reduce the processing search speed. However, for a completely connected graph, circuits exist. Since circuits are unavoidable, our graph design ensures that circuits are as long as possible (since this improves search speed by making useful paths shorter). Our system includes a small working memory that keeps track of the next best single node previously checked but not chosen and the next best initial input node. This allows us to achieve backtracking by perturbation jumps to these nodes.

IV. Directed Graph Autonomous Synthesis

We now detail the design and construction of our directed graph. We note that the procedure is automatic and does not require human intervention. The first issue is to select the out-degree M of each node. This is a function of the number of parallel processor channels available M_1 (i.e., how many adjacent nodes can we check in parallel). If $M = 1$, the search through the graph is sequential, and for a graph with N nodes (with the time for one comparison at a node being T), the search time through the graph is $O(NT)$, where the operator O denotes order of. If we search M nodes in parallel at each level, one can show that

$$\text{search time} = O[(1/\gamma)T \log_M N] = O(T \log_M \gamma N), \quad (2)$$

where $\gamma \leq 1$ is the graph efficiency. (This measures the search time of a graph design with respect to the optimal graph in which every node can be reached by exactly one path of length $\log_m N$ or less.) In practice, one cannot easily achieve such optimal performance and should not expect γ values above 0.5. We expected and found γ to decrease as N increased and that graphs with short circuits had poorer (lower) γ values. For our case study, the optimal search time is $2.74T$, and our design gave an average of $5.27T$ for a quite good graph efficiency $\gamma = 0.52$. Without our *meta*-starting vertices, the average search time was much worse ($8.5T$).

We assume a parallel processor with M_1 channels (capable of performing M_1 vector comparisons in parallel in a time T), one can show that if $M_1 = 1$, then $M = 2$ yields the minimum search time; and if M_1 is >1, the minimum search time is obtained when we choose $M = M_1$ (i.e., when the out-degree per node in the graph matches the parallelism of the processor being used). In the multiprocessor case, the number of nodes to be searched is still $O[(1/\gamma) \log_M N]$, and the search time given by Eq. (2) is reduced as M increases. A similar analysis shows that the same value of M also represents the optimal number of starting or initial nodes.

We now discuss how the M nearest neighbors of a node are selected during synthesis of the graph (i.e., how the M nonzero entries per row of the adjacency matrix A_1 are selected). This is achieved by forming the VIP matrix R for the N input data vectors x_i, normalizing R to produce unit diagonal elements. (This keeps R positive-definite and prevents a vector with a high magnitude from dominating comparisons.)[11] Entry a_{ij} (with $i \neq j$) in A_i is 1 if element r_{ij} of the normalized R is one of the M largest elements and is zero otherwise. Subsequent adjacency matrices A_n and the extent matrix E can then be obtained using Eq. (1). We use R to determine outgoing arcs only.

We found that a more detailed procedure is necessary to assign incoming nodes and that if R is used for this, the graph will have many subgraphs and not be well connected. The procedure we use to determine which nodes have incoming arcs to the node in equation is detailed elsewhere[4] and is thus only highlighted below. We form matrices R′ and A_1' that are reduced versions of R and A_1 containing only the largest elements of R and the columns of A with nonzero entries. (This reduces storage by a factor of N/M.) We also form an $N + 1$ element working vector array z whose elements are the comparisons of the input vector x (to be added to the graph and knowledge base) with the previously chosen input class vectors. We note that the outgoing connections using R have established some ingoing arcs into some nodes. For the node associated with the newest input vector, we check all prior nodes to see which have high-comparison agreement with the new node. We also check all prior nodes to see which outgoing arc from them has the smallest weight or agreement with another node. We check

whether the prior nodes which have high comparison agreement with the new node are also connected to other nodes, which also have high agreement with the new node in question. Simple manipulations of z, A_1', and R′ allow us to determine which previous nodes should have arcs into the new node and which old arcs to break. When an old arc is broken, we reroute the old connection through the new node. The new A_n matrices are used to indicate the reachable extent of nodes whose arcs have been altered. The rows of R′ and A_1' are now recorded with these new data. If no ingoing arcs to the new node result (i.e., if arcs to be broken to allow arcs into the new node have greater strength than the new arc added), we force a connection into the new node. (This is essential to maintain the connectedness of the graph.) We do this by adding an arc from the new node to the node (node *A*) with the largest agreement with the new node and breaking the weakest arc from node *A* (which we assume goes to node *B*). We then insure that node *B* is connected to the graph. (This is generally the case since the new node will have node *B* as one of its adjacent nodes.)

An example will best demonstrate several aspects of the above procedure. In Fig. 4, we show the graph that resulted when the first five letters (*A*–*E*) were shown to the system. We restricted this graph to $M = 4$ outgoing arcs per node. When the sixth input letter *F* was fed to the system, the graph that resulted is shown in Fig. 5. The modifications produced are automatic and follow the rules noted above. We now discuss and compare the graph structures in Figs. 4 and 5 to gain insight. We note that node *F* is connected (by outgoing arcs) to nodes *A*, *B*, *E*, as well as *D*. Visual comparison of the letters shows that these are the four letters to which *F* is most similar. Let us now consider the ingoing arcs into node *F* and how they are determined. We note that in Fig. 4 an arc from *E* to *D* existed and that in Fig. 5 this arc is broken and replaced by arcs from *E* to *F* and *F* to *D*. This choice was made since *F* is more similar to *D* than is *E*, and *F* is more similar to *E* than to *D*. The arcs from *A* to *D* and from *B* to *D* in Fig. 4 are also rerouted through node *F* as seen in Fig. 5.

During synthesis of the graph (and during its use in classification), if a new input class is sufficiently close to a previously introduced data vector or class vector, the new input class can be assigned to a prior node or the threshold can be increased and a new node added (as described above). This rule is under control of the user. We now address how to restructure the graph. This arises in the design of the graph and in the use of the graph during classification of input data and knowledge. This case arises when we have reached the maximum number of nodes *N* allowed in the graph (e.g., for memory reasons, we cannot add another node, but yet the input data do not agree sufficiently with the vector data at any existing node). In this case, we must merge two nodes of the graph into one node, rearrange their interconnections, and add a new node (the new input vector). This is easily achieved by reducing the threshold and from the reduced size R′ matrix noting which node to merge with the node with

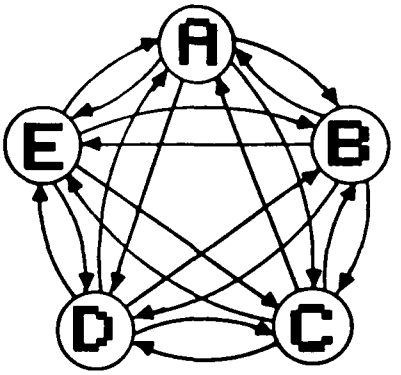


Fig. 4. Directed graph for the first five characters in the alphabet.

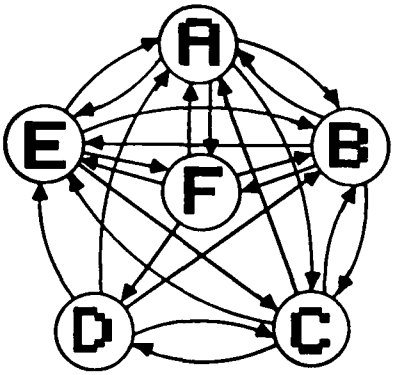


Fig. 5. Directed graph when *F* the sixth letter of the alphabet is added to the knowledge base.

the lowest VIP value. (This node vector is stored in the first column of R′ in the ordering procedure we employ.) A new data vector representative of the two nodes to be merged is formed by averaging their vectors. All arcs into these two nodes are broken and replaced by arcs to other nodes (those with the largest R′ entries). This is equivalent to removing two columns from A and R. We also remove all outputs arcs from these nodes. (These are determined from the rows of A_1.) This removes two rows of A and R. After the aforementioned procedure, these nodes have now been removed from A_1, R and from the graph itself. The merged node is then added to the graph as a new node using the techniques noted above and demonstrated in Figs. 4 and 5.

V. Optical Realization

We now discuss the optical realization of a directed graph processor. We note that small working memories are sufficient to handle the updating and restructuring of the graph. We note that these memory matrices have either binary entries or analog entries (that do not require high accuracy) because of the nature of our search procedure and the interconnectedness of the graph (which allows errors at one node to be subsequently corrected later since a given node can be reached by many paths). Thus these operations should be able to be implemented with simple analog

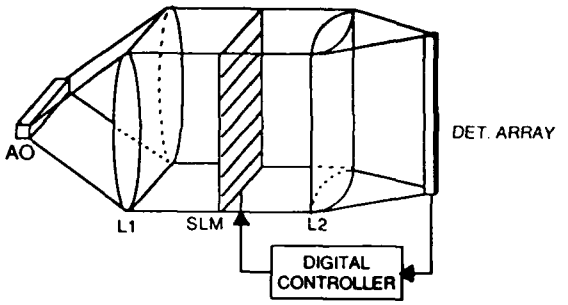


Fig. 6. Multichannel space-integrating parallel optical directed graph processor.

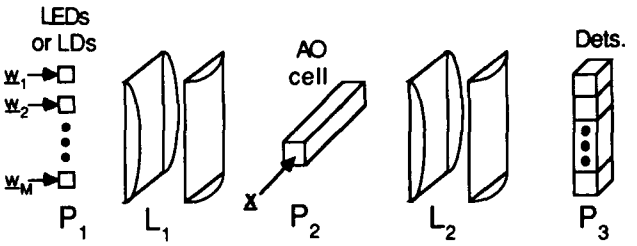


Fig. 7. Multichannel time-integrating parallel directed graph processor.

optical matrix processors.[12] Since the directed graph has many interconnections (outgoing arcs from each node), it is most suitable for an optical realization, since optical systems can easily achieve broadcasting (one-to-many connections) and massive interconnections with no crosstalk between the connection lines.[13,14]

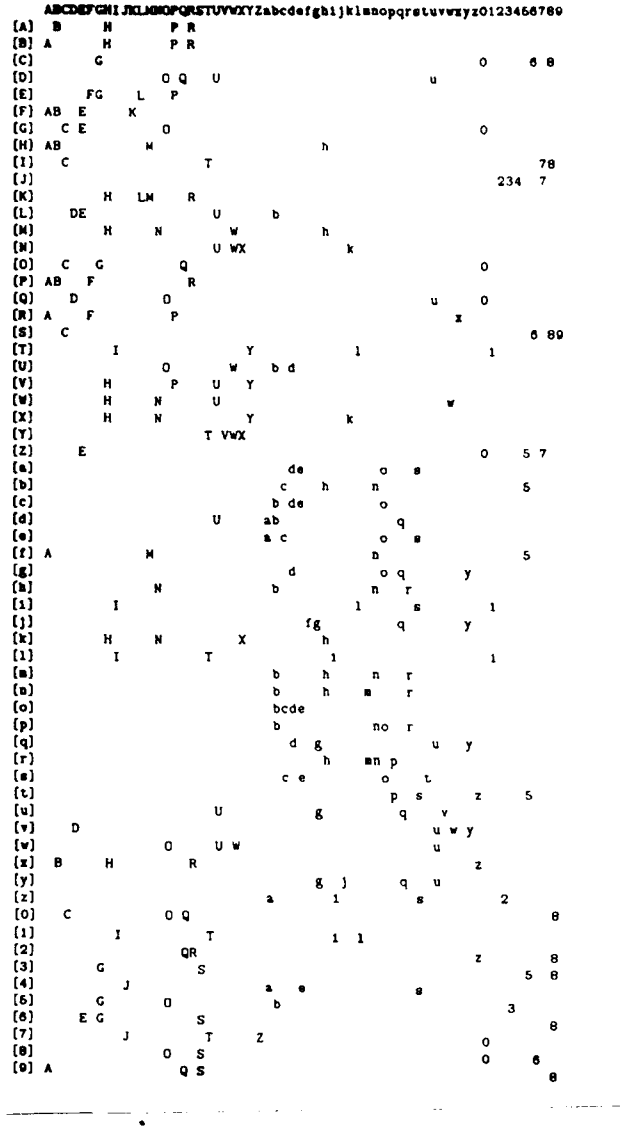
Figure 6 shows one realization of an M-channel parallel processor system to perform the M comparisons necessary per node in the directed graph. In Fig. 6, the input vector to be classified is fed to an acoustooptic (AO) cell, and the M vectors corresponding to the nodes adjacent to the node being searched are entered into the M rows of a 2-D spatial light modulator (SLM), preferably an M-channel AO cell. The contents of the input AO cell are broadcast in parallel to the M rows of a SLM by lens $L1$. They pass through the SLM forming the point-by-point product of the elements of the input vector and the M reference vectors. Lens $L2$ then sums each product onto a separate detector in the output plane. The M output detectors thus contain the M vector inner products of the input data and the M outgoing node data vectors. By monitoring the time history output from the detectors, the system can achieve correlations of the input vector with each of M reference vectors. (This can be a shift-invariant comparison of the vectors, should this be required.) The AO cell can be replaced by a linear array of LEDs or laser diodes (LDs) (should the vector data be available in parallel).

A multichannel time integrating correlator can also be used (see Fig. 7) if the vector data are long, or if its data rate is slow. In this well-known architecture, the M reference vectors are fed to separate point modulators (LDs or LEDs) at P_1 and the vector x to be classified is fed to an AO cell at P_2. This light leaving each point modulator is broadcast to illuminate uniformly P_2 with different angles of incidence. Plane P_2 is imaged onto plane P_3. Since the data from each P_1 point modulator enter P_2 at a different angle, it appears at a different vertical location in P_3. Thus the product of each P_1 input and the contents of P_2 appear on separate P_3 detectors. These detectors time integrate this input signal data for the duration of the input signals (vectors). The P_3 outputs are the correlation of the x and the M references w_m on the M output detectors. In the case of symbolic vectors or feature vectors, shift invariance is not required, and M vectors inner product comparisons are sufficient. In this case, we only require one detector per row at P_3 (rather than a linear detector array at each vertical location in P_3), since the alignment of the vectors is assumed to be known.

A final architecture to implement a directed graph processor could employ optical (holographic etc.) storage of the M outgoing node vectors associated with each of the N nodes in the graph (in various space and frequency multiplexing arrangements). One could access the appropriate M data vectors with an optical beam deflector and perform the comparison directly, or one could readout these M vectors and write them on an optically addressed material in the plane of the 2-D SLM in Fig. 6. Optically addressed SLMs and materials for optical storage are not as developed and mature as are LEDs, LDs, and AO devices. Thus we restrict our present attention to the architectures of Figs. 6 and 7.

We now discuss the search time possible with these architectures. We consider a sixteen-channel AO cell for the SLM in Figs. 6 and 16 laser diodes at P_1 in Fig. 7. Thus we consider a directed graph with $M = 16$ parallel processors or outgoing arcs per node. We assume a quite modest 1 MHz = 1-Mbit/s input data rate to each AO cell and to the laser diodes. We consider the case of fifty-element vectors and even include the fact that they will be encoded to eight-bit accuracy; i.e., each vector will be a $50 \times 8 = 400$-element binary vector. In this case, new vectors can be input each 0.4 ms, and the processing time required for parallel comparisons per node will be $T = 0.4$ ms. This length T necessitates use of the time integrating architecture of Fig. 7. We consider a knowledge base with $N = 2^{12} = 4096$ classes. With a graph efficiency of one, the average time for classification of the input data (the search time for this knowledge base) is $O(T \log_M N) = 1.2$ ms, and the digital memory requirements are modest (only ~0.5-Mbits). Thus, even with very modest device requirements, the architectures of Figs. 6 and 7 are quite attractive. AO cells with sixty-four channels and LD arrays with sixty-four elements are realistic, as are data rates of 1 GHz; thus very fast searches of very large knowledge bases are possible with this architecture.

Table II. Adjacency Matrix A for the $N = 62$ Node Alphanumeric Case Study Directed Graph



VI. Case Study

We now summarize the results of our directed graph $N = 62$ class alphanumeric case study with $M = 4$ outgoing nodes assumed and with sixty-four-element iconic binary input vectors used. The graph was assembled with one input vector at a time using a threshold of 0.99. Table II shows the adjacency matrix A_1 that describes the graph. We note that synthesis of the graph was automatic (following the rules highlighted in Sec. IV). Each row of the matrix shows the adjacent nearest neighbor nodes for the node associated with the character indicated in the left margin. The *meta*-vertices used were shown earlier in Fig. 3. The directed graph was able to classify all sixty-two inputs perfectly. Noise tests were not performed, or were tests employing font and point size variations, since our present purpose was to demonstrate the design of the graph and the organization of such a knowledge base and to quantify the efficiency of the graph designed. As noted earlier, the optimum number of nodes that a graph would examine (on the average) to search the full knowledge base is $\log_M N = 2.74$ for our case. We calculated the number of nodes searched for all sixty-two inputs (the cardinality of the path from entering the graph to the final node selected) and found that our graph required an average of only 5.27 node searches to classify all sixty-two input vectors. This represents a most respectable graph efficiency $\gamma = 0.52$.

VII. Summary and Conclusion

We have advanced an autonomous technique to organize a knowledge base and allow an efficient search through a large knowledge base. Borrowing and modifying techniques from directed graphs, we devised automatic and efficient procedures to determine the interconnections of the nodes, and new techniques to add new nodes to the graph and restructure the graph as new knowledge are learned. This provides us with a technique that adapts to new knowledge and automatically restructures its knowledge as new data enter. Data classes are stored as nodes, and associations are stored as arcs or interconnections. The organization proposed allows use of parallel multiple processors to reduce search time by employing many interconnections. It is thus quite suitable for optical implementation. This graph technique is preferable to tree structures and hierarchical relational graph processors (by virtue of its use of interconnections, parallel processors, and the ability to reach any node from any node by a number of paths).

We detailed several optical designs for this adaptive learning processor. We quantified its efficiency for a case study, and quantified how large data bases can be searched quite fast employing it. Our case study showed the ability of such a knowledge base processor to classify correctly all sixty-two upper and lower case alphanumeric characters.

The support of this research by an optical neural processing grant from the Air Force Office of Scientific Research is gratefully acknowledged and appreciated.

References

1. *Technical Digest, Topical Meeting on Optical Computing* (Optical Society of America, Washington, DC, 1985).
2. *Technical Digest, Topical Meeting on Optical Computing* (Optical Society of America, Washington, DC, 1987).
3. N. Christofides, *Graph Theory: An Algorithmic Approach* (Academic, New York, 1975).
4. E. Baranoski and D. Casasent, "A Directed Graph Optical Processor," Proc. Soc. Photo-Opt. Instrum. Eng. **752**, 58 (1987).
5. P. H. Swain and H. Hauska, "The Decision Tree Classifier: Design and Potential," IEEE Trans. Geosci. Electron. **GE-15**, 142 (1977).
6. E. M. Rounds, "A Combined Nonparametric Approach to Feature Selection and Binary Decision Tree Design," Pattern Recognition **12**, 313 (1980).
7. I. K. Sethi and G. P. R. Sarvarayudo, "Hierarchical Classifier

Design Using Mutual Information," IEEE Trans. Pattern Anal. Machine Intell. **PAMI-4,** 441 (1982).
8. D. A. Jared and D. J. Ennis, "Learned Pattern Recognition Using Synthetic-Discriminant-Functions," Proc. Soc. Photo-Opt. Instrum. Eng. **638,** 91 (1986).
9. H. S. Stone and P. Sipala, "The Average Complexity of Depth-First Search with Backtracking and Cutoff," IBM J. Res. Dev. **30,** 242 (1986).
10. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," Science **220,** 671 (1983).
11. J. M. Wozencraft and I. M. Jacobs, Principles of Communication Engineering (Wiley, New York, 1965).
12. Special Issue on Optical Computing, Proc. IEEE **72** (July 1984).
13. J. W. Goodman, F. J. Leonberger, S-Y. Kung,and R. A. Athale, "Optical Interconnections for VLSI Systems," Proc. IEEE **72,** 850 (1984).
14. Special Issue on Optical Interconnections, Opt. Eng. **25,** No. 10 (Oct. 1986).

CHAPTER 6:

"MULTIPLE DIRECTED GRAPH LARGE-CLASS MULTI-SPECTRAL PROCESSOR"

Proc. SPIE
Volume 974
August 1988

Multiple Directed Graph Large-Class Multi-Spectral Processor

David Casasent, Shiaw-Dong Liu and Hideyuki Yoneyama
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

ABSTRACT

The use of a decision net (hierarchical classifier and a multiple directed graph processor) is detailed and demonstrated for an imaging spectrometer identification problem.

1. Introduction

Section 2 briefly reviews the imaging spectrometer problem we consider. Section 3 reviews our hierarchical classifier with its automated and not ad hoc tree structure and linear discriminant function (LDF) design. Section 4 presents our new multiple directed graphs and new directed graph design concepts. Section 5 presents our initial data. Our conclusions are advanced in section 6.

2. Imaging Spectrometer Processing Problem Definition

Imaging spectrometer sensors such as the airbone imaging spectrometer (AIS), the airbone visible/infrared imaging spectrometer (AVIRIS), and the planned high resolution imaging spectrometer (HIRIS) systems[1-3] offer the ability to provide spectral reflectance data (responses at wavelength λ_l, l=1, .., L) for each (x,y) image pixel. This 3-D (x,y,λ) data rate can approach 1 Gbyte/sec with the number of operations per second being much higher. We use L= 128 wavelength reflectances for E=500 mineral elements as our reference database (these include different color and grain size samples per mineral as separate classes in our E= 500 class reference database). For cases when the recorded spectra are mixtures of several spectra, a neural net processor is required[4-6]. This paper considers cases when the wavelength response k^e of a reference element e for different wavelengths l is available as reference data to process the received signal at each pixel. The problem is to determine which element e is present in the input data from the input spectrum, when one element e is assumed to be presented in each input imaging spectrometer pixel.

3. Hierarchical Classifier

To determine the class of an unknown input, we must compare its spectrum with the E= 500 references. We consider this as a large class problem or large knowledge base search for which we now

review our previously detailed[7] hierarchical classifier. This consists of a binary tree with P_c= 100% correct classification with high confidence (measured by the Fisher ratio of the designed LDF at each node). We consider a binary classifier since it provides better performance[8]. To provide an automatic design of the structure of the classifier, we use a new iterative Fisher/minimum-variance algorithm[7] to select the two clusters of the data to be used at each node. In this algorithm, we first perform a minimum-variance clustering of the data at each node (e.g. E=500 classes at node 1 of Figure 1). We then compute the Fisher LDF for these initial clusters, project the data onto this Fisher LDF, and measure its P_c for the training set data. If P_c is not 100%, we perform minimum-variance clustering (in Fisher projection space now) to obtain a new partition of the training data. We then calculate a new Fisher LDF for this new pair of clusters and compute P_c for it. This iterative procedure is repeated until P_c= 100% is obtained. The Fisher ratio increases (and hence our confidence) at successive iterations. This iterative procedure is performed for each node in the tree. Thus, this algorithm maximizes the Fisher ratio (our high confidence performance measure) and provides P_c= 100% (which a Fisher LDF does not normally achieve). The design tree structure (the best pair of clusters to be separated per node) is automated as is the design of the Fisher LDF used at each node. At successive nodes, the number of wavelengths L used is reduced to allow the inversion of the covariance matrix in the calculation of the Fisher LDF. We terminate the hierarchical classifier after 4 levels (as shown in Figure 2) either because the Fisher ratio (our confidence measure) is small or because the number of classes at that node (and thus the number of features used) is few. Figure 2 shows the number of classes at each node and the Fisher ratio for the LDF at each node. As shown, this hierarchical classifier yields 7 final clusters (with 30-155 elements per cluster). We plan to use this hierarchical classifier to feed several directed graphs (one for each of the seven clusters) to determine the class of the input data (Figure 3). Our data presented here used unnormalized data to produce the data clusters (hierarchical classifier) and zero-mean and normalized data for the VIP per node in the directed graphs.

4. New Multiple Directed Graph Design

Directed graphs[9] (e.g. Figure 4) are quite useful in organizing and searching a large knowledge base.[10]. Since they have many interconnections (with outdegree M per node), they are highly interconnected and can be designed to allow any node (class) to be reached from any other node (thus allowing errors obtained in earlier classifications to be overcomed without the need for backtracking with its associated problems). Thus directed graphs allow problems[11] in conventional sequential classifiers[12-14] to be overcome.

Our new P_c= 100% high confidence hierarchical classifier (for clustering only) and the use of multiple directed graphs for final class decisions combine the best features of both into a new decision net for large class problems. Directed graphs allow new nodes to be added to the graph and the graph to be restructured and are thus also adaptive. However, the standard directed graph design techniques[9-10] have several shortcomings. These include: many connections are forced (to improve the connectivity of the graph), these forced nodes and nodes with few ingoing arcs resulted in local maximum, which require perturbations in the graph search and full connectivity is still not achieved, and different directed graphs result for a different order with which the training set vectors are

presented.

Our new directed graph design overcomes these problems with new graph design rules. These rules include: use of only natural arc connections obtained from the VIP matrix (this reduced the number of perturbations required and improves the graph search time), use of multiple graphs (whose clusters are obtained from our hierachical classifier), and new starting node rules. The need for new starting node rules arises since the use of only natural arcs results in disjoint subgraphs (where a subgraph is a collection of nodes which one cannot enter and leave), fully connected or not (where a fully connected subgraph allows any node to be reached from any other node in the graph). These subgraphs can easily be located from the extent matrix. Several subgraphs can exist within each of the multiple graphs. We use the hierarchical classifier to determine the elements to cluster within each multiple graph. We must also determine which starting nodes to enter in each graph. Nodes with a large number of ingoing arcs will be visited often, while nodes with few ingoing arcs will be visited seldom. Thus, the input corresponding to such seldom visited nodes are those for which we expect to encounter local maximum since there is only one or a few paths (arcs) into the proper final node. Local maxima are nodes for which a test input yields smaller VIPs with all M neighbors of the present node than the VIP at the present node. In such cases, we perturb the directed graph. Thus, we use nodes with few ingoing arcs as starting nodes to reduce the number of local maxima that occur. Local maxima occur in all directed graphs (even in our new algorithm). One can locate these only by exhastive test and it is unrealistic to try to eliminate all of them. Our use of only natural arc connections reduces the number of local maxima and our starting node selection rules reduces the problem of encountering local maxima. Our use of VIP data (and natural, not forced arcs) to design the graph results in the same graph design regardless of the ordering of the training set data.

The connectivity and reachability will be worse with our graph design since disjoint subgraphs result. We solve this by using our hierarchical classifier to determine the clusters for the multiple graphs and by using multiple starting nodes. We determine which graph to enter using the hierarchical classifier. We determine how to enter the graph for each cluster using new starting node rules (to ensure full reachability and a fast search of the directed graphs). Our new graph design procedure produces fewer local maxima and fewer perturbations (to reduce the average search time). Our use of multiple graphs and new starting node rules is attractive since as the size of any graph grows, its efficiency γ decreases. In determining the class (node assignment) of input test data, we first perform VIPs of the input vector with the LDFs at different nodes. We could also use this to determine which of the multiple directed graphs to enter. We then perform VIPs with the data at the starting nodes of that graph, select the node with the largest VIP output and then perform VIPs with its M neighbors. We continue this process until the VIP at the current node node is larger than the VIPs with its M neighbors and then we check to see if the VIP exceeds our threshold τ. If not, we perturb or jump to the previously checked (but not used) node with the largest VIP value (this data is easily stored). The threshold τ is selected as halfway between the VIP with the correct node (always 1) and the next largest VIP in the VIP matrix. We use an outdegree of M for all nodes (where M is the number of parallel or multiple processors available). Similarily, we use a number of starting nodes that is an integer multiple of M.

We now detail our starting rules. These include: select one starting node per subgraph, pick nodes with the fewest ingoing arcs (these correspond to inputs with the largest probability of being trapped

in a local maximum), pick nodes that reach the most nodes (thus larger subgraphs generally posseses more starting nodes than smaller ones), pick additional starting nodes such that they reach nodes not reached from the initial starting nodes, and pick the nodes which are hardest to be reached from the initial starting nodes. In all of our standard and new directed graphs, we perturb by jumping to the node (previously checked) with the largest VIP. This procedure does not always (by luck) yield the best performance, but it is a rational criterion. An alternative choice for perturbations is to jump to a node in an unused region of the graph (this is a topic for future research).

To add a new node or restructure (without increasing the number of nodes) a graph (this is necessary if no VIPs with the input data exceed the threshold τ), we use the new VIP matrix (it has only one additive row and column) to combine two nodes (those with the largest VIP with each other) and then add the present input as a new node (class). We generally merge two nodes whose VIPs are close to each other to improve noise performance. These graph design and starting node rules are easily controlled by the VIP, adjacency, and extent matrices.

The starting nodes may not need to be altered when a new node is added or when the directed graph is restructured. When adding a node, we look at the added node and all nodes with broken ingoing arcs and the reachability of these nodes. If the reachability is adequate, we make no change in the prior starting nodes. When the VIP entries for the new node (the new VIP column and row) are larger than some of the M arcs used already, we must include the new node as the end of one of the M arcs from a prior node. We do this by breaking old weak arcs and adding the new arcs needed.

5. Test Data

In all of our tests of standard and our new directed graphs, the same perturbation rules were used (i.e. we jump to the previously checked node with the largest VIP). For our new directed graphs, the starting nodes used were chosen as noted in Section 4. For the starting nodes for the standard directed graph, we calculated the total number of perturbations that would be required (for all data in the graph) for each possible starting node and we selected the starting nodes as the nodes that required the fewest perturbations. In all cases, P_c=100% recognition was obtained.

Table 1 shows the performance of the standard directed graph (one directed graph for the E=500 class problem) for two different choices of M, using both standard and new (without the multiple graph structure) direct graph design algorithms. Their efficiencies are low since many perturbations are required. From this, we see that the use of our new graph design rules (without multiple graphs) produces an improvement in search time (of 20-27%) and in graph efficiency (of 17%-38%). If we solve the same problem with seven separate directed graphs (using standard graphs and our new graph rules), we obtained an improvement in search time by a factor of 5 (Table 2). Most of the improvement is thus due to the use of multiple graphs. (Note, however, that the exhaustive starting node search used for the standard directed graph is not suitable for very large class problems and when the directed graph is changed adaptively). In Table 3, we show the average search time for each of the seven standard and new directed graph rules for the same set of seven clusters. As seen, our new directed graph rules improved performances in general (up to a maximum of 33%), with a general (and encouraging) trend to better performance for larger directed graphs (e.g. cluster C155), and with poorer performance (by luck) for several cases. Table 4 lists the graph efficiencies for the seven

directed graphs in Table 3 using both algorithms. As seen, our new algorithm performs better than the standard one or nearly the same for all cases with a general trend to better performance for clusters with more elements. Note that the graph efficiency γ measures how close the graph comes to the ideal average search time for N nodes, $O[(1/\gamma)\log_M N]$ where the ideal $\gamma=1$ is never achieved. The threshold value used in all data tests was $\tau=0.99$. The value is high, but has a valid range when noise is not present.

Table 5 lists various parameters from our seven new directed graphs in Tables 3 and 4. For M=4, the number of starting nodes is always less than the number of fully connected subgraphs (and is an integer multiple of 4). For M=8, the number of starting nodes is always 8. We do not need 8 starting nodes to make all the nodes reachable, but we use 8 to fully utilize the parallelism in the processor (M is the number of parallel processors available). We do not need to choose one starting node for each fully connected subgraph in a cluster to make all nodes in a cluster reachable. Note that several nodes are in several fully connected subgraphs. This can be seen from the fact that there are more fully connected subgraphs (each node can be reached from any node) than standard subgraphs (no path both in and out). In all the clusters, all nodes are reachable from one of the starting nodes.

In the standard directed graphs design, the rule we used when adding a new node and ingoing arcs to it was as follows. We added outgoing arcs to the M nodes (in the present graph, not in the full graph) with the largest VIPs. We added ingoing arcs to the new node from nodes whose VIPs (with the new node) are larger than one or more of their prior neighbors. These are referred to as natural arcs. However, since all nodes have only M outgoing arcs, we must break a prior arc to provide an ingoing arc to the new node. This was done only if the node with the broken ingoing arc can still be reached from the node with the broken outgoing arc through some other path (i.e. the graph must remain fully connected). If no such natural ingoing arcs can be made, then a forced arc is necessary (i.e. the new node has no ingoing arcs). If there is a direct connection (of length one) between the nearest neighbors of the new node, then we break the weakest such connection and connect that outgoing arc to the new node. If no such connection exists, then none is made and a fully connected graph does not result (at this stage of graph synthesis). A major problem with standard directed graph design is that the training node data is fed sequentially to the directed graph synthesis stage and that we must force arcs to maintain connectivity. However, we can only connect to the nodes present thus far. Later in graph synthesis, new nodes are added and these have natural arcs into a prior node with a forced ingoing arc. In this case, the natural arc that was broken to force an earlier ingoing arc should and could often be replaced (but the bookkeeping for this is not realistic for a large database).

In other data for the standard single E=500 (M=4) directed graph, we found 64 forced arcs (64 nodes with forced ingoing arcs). For those nodes with a forced ingoing arc, we found 23 nodes with one final ingoing arc, 17 nodes with 2, 13 nodes with 3, and 11 nodes with 4 to 6 final ingoing arcs. For the standard single E=500 M=8 directed graph, we found 26 forced arcs. Of the 26 nodes with forced ingoing arcs, 10 nodes had one final ingoing arc, 2 had 2 ingoing arcs, and 6 had 3-5 ingoing arcs. By comparison, our new graph construction yields more nodes with only 1, 2 etc. ingoing arcs (and several nodes with no ingoing arcs). However, our use of several starting nodes solves these problems for our directed graphs. (We still employ M starting nodes in our comparisions of the standard and our new directed graphs). For the two standard M=4 and 8 directed graphs for the

E=500 class problem, we measured the search time for the 23 and 10 inputs corresponding to nodes with only one ingoing arc and compared this to the average search time for each graph, we found a search time of 129 v.s. 66 and 92 v.s. 33 for the M=4 and 8 cases. Standard starting nodes selection (described earlier) and perturbation rules (perturb to the previously checked node with the largest VIP) were employed. These data show that these nodes with few ingoing arcs definitely require longer search time (by factors of 2 to 3), and thus our new starting nodes, multiple graphs, and graph design techniques should be most attractive. The fact that our graph design results in the same directed graph for all orders of training vectors is a vital issue, whose importance is not apparent from the present data (since all various possible directed graphs with different orders of training vectors were not tested). Our new directed graph is better, primarily due to the use of multiple graphs and our choice of starting nodes (not due to the graph design itself).

6. Summary and Conclusions

A new clustering method for the design of a hierarchical classifier has been reviewed. Its use with multiple directed graphs represents a new and efficient decision net. We select the clusters for the multiple graphs from this hierarchical classifier and use the hierarchical classifier to determine which directed graph to enter. New directed graph rules (to reduce local maxima and number of perturbations required) and new starting node rules (to extend the reachability and to reduce the seach time of the graphs) were advanced. Initial test results on a 500 class imaging spectrometer problem were very successful with P_c=100% and an improvement factor of over 5.

Acknowledgement

The support of AFOSR for our directed graph work and JPL for the imaging spectrometer application is gratefully appreciated.

References

1. R. MacDonald, "A Summary of the History of the Devolopment of Automated Remote Sensing for Agriculture Applications", *IEEE Trans. Geoscience and Remote Sensing*, Vol. GE-22, Nov. 1984, pp. 473-481.

2. G. Vane, A. Goetz, and J. Wellman, "Airborne Imaging Spectrometer : A New Tool for Remote-sensing", *IEEE Trans. Geoscience and Remote Sensing*, Vol. GE-22, No. 6, Nov. 1984, pp. 546-549.

3. A. Goetz, G. Vane, J. Solomon, and B. Rock, "Imaging Spectrometry for Earth Remote Sensing", *Science*, Vol. 228, Jun. 1985, pp. 1147-53.

4. E. Barnard and D. Casasent, "Optical Neural Net for Classifying Image-Spectrometer Data", *Applied Optics*, (submitted) 1988.

5. E. Barnard and D. Casasent, "New Optical Neural System Architecture and Applications", *Proc. SPIE*, Vol. 963, Sept. 1988.

6. E. Barnard and D. Casasent, "New Tracking and Matrix Inversion Optical Neural Nets", *IEEE ICNN*, San Diego, July 1988 (Poster Paper).

7. Shiaw-Dong Liu and David Casasent, "Optical Processing of Spectrometer Data", *Proc. SPIE*, Vol. 938, Apr. 1988.

8. B. Kim and S. Park, "Comments on 'An Automated Approach to the Design of Decision Tree Classifiers", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 4, July 1986, pp. 500.

9. N. Christofides, *Graph Theory, an Algorithmic Approach,* Academic Press, New York, 1975.

10. D. Casasent and E. Baranoski, "Directed Graph for Adaptive Organization and Learning of a Knowledge Base", *Applied Optics*, Vol. 27, Feb. 1988, pp. 534-540.

11. H. Stone and P. Sipali, "The Average Complexity of Depth- First Search with Back- Tracking and Cutoff", *IBM J. Res. Develop.*, Vol. 30, No. 3, May 1986, pp. 242-258.

12. P. Swain and H. Hauska, "The Decision Tree Classifier: Design and Potential", *IEEE Trans. Geosci. Electron.*, Vol. GE-15, No. 3, July 1977, pp. 142-147.

13. E. Rounds, "A Combined Nonparametric Approach to Feature Selection and Binary Decision Tree Design", *Pattern Recognition*, Vol. 12, 1980, pp. 313- 325.

14. I. Sethi and G. Sarvarayudo, "Hierachical Classifier Design Using Mutual Information", *IEEE Trans. Pattern Analy. Machine Intell.*, Vol. PAMI-4, Sept. 1982, pp. 441-452.

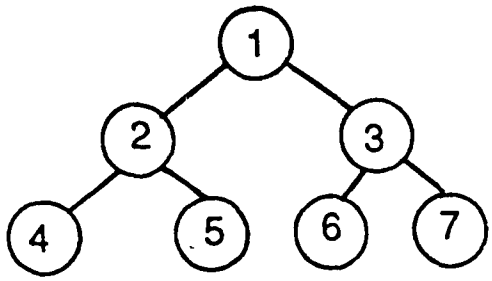


Figure 1: Typical hierarchical classifier

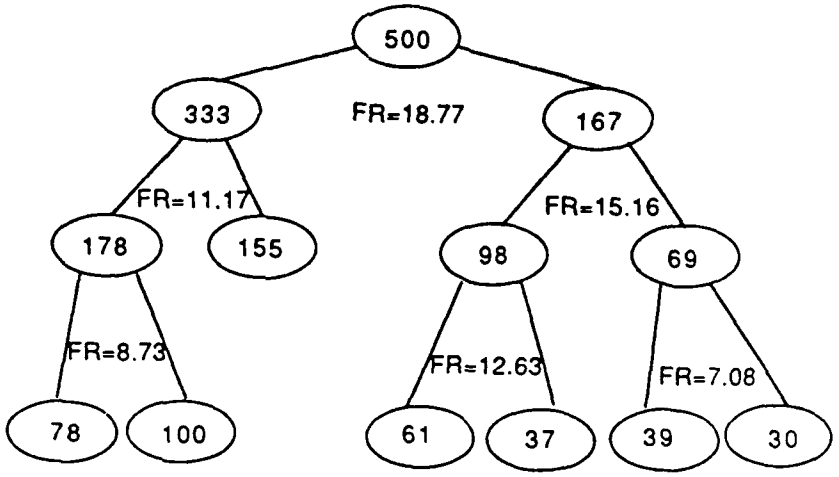


Figure 2: Our resultant hierarchical classifier for E=500 classes

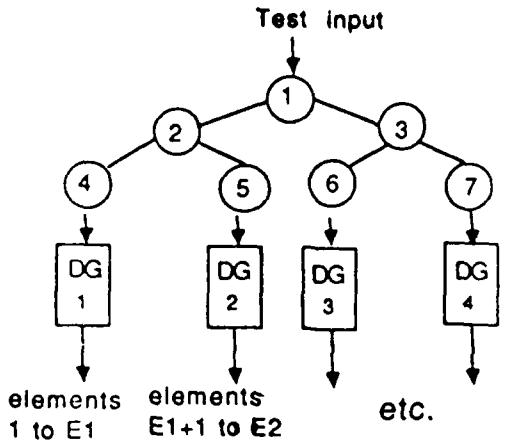


Figure 3: Decision net (hierarchical classifier feeding multiple directed graphs)

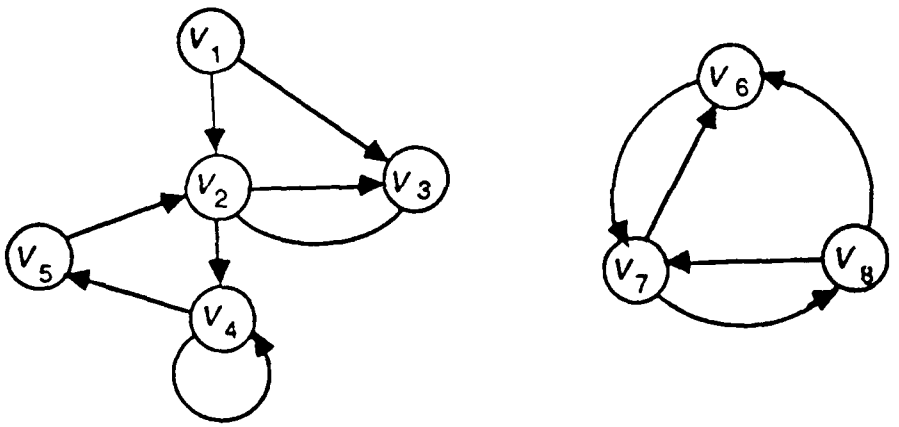


Figure 4: Typical directed graph

alg. / M	Average search time		Efficiencies (γ)	
	Standard	New	Standard	New
4	65.7	50.12	0.08	0.11
8	32.7	27.78	0.12	0.14

Table 1: Average search time and efficiencies of the standard graph and new directed graph (without the multiple graph structure) using only one graph for the E=500 class problem.

alg. / M	One graph		Multiple graph	
	Standard	New	Standard	New
4	65.7	50.12	12.53	9.54
8	32.7	27.78	5.16	4.58

Table 2: Average search time for one graph vs multiple graphs using standard and new algorithms.

M / cluster / alg.		Average search time of the seven directed graphs						
		C30	C37	C39	C61	C78	C100	C155
4	Standard alg.	3.43	4.08	4.10	6.26	18.77	14.01	16.81
	New alg.	3.57	4.09	3.85	5.20	6.53	13.96	11.92
8	Standard alg.	2.93	3.22	2.92	3.66	4.05	5.96	6.62
	New alg.	2.73	2.78	2.79	3.31	3.78	4.88	6.53

Table 3: Average search time of the seven graphs using both standard and new algorithms. The directed graphs are denote by cluster (c) followed by the No. of elements per cluster.

M \ cluster / alg.		Efficiencies of the seven directed graphs						
		C30	C37	C39	C61	C78	C100	C155
4	Standard alg.	0.99	0.85	0.85	0.56	0.17	0.25	0.23
	New alg.	0.99	0.85	0.92	0.69	0.57	0.25	0.33
8	Standard alg.	0.84	0.78	0.99	0.74	0.60	0.45	0.44
	New alg.	0.95	0.97	0.99	0.85	0.74	0.57	0.45

Table 4: Graph efficiencies of the 7 graphs in Table 3 tests using both standard and new algorithms

M	Numbers	C30	C37	C39	C61	C78	C100	C155
4	No. of starting nodes	4	4	4	8	8	4	12
	No. of subgraphs	1	1	2	1	1	1	1
	No. of fully connected subgraphs	1	7	4	11	10	24	23
8	No. of starting nodes	8	8	8	8	8	8	8
	No. of subgraphs	1	1	1	1	1	1	1
	No. of fully connected subgraphs	1	2	1	4	1	9	7

Table 5: Number of starting nodes, subgraphs, and fully connected subgraphs in the 7 directed graphs using our new design algorithm.

CHAPTER 7:

"KEY AND RECOLLECTION VECTOR EFFECTS ON HETEROASSOCIATIVE MEMORY PERFORMANCE"

Key and recollection vector effects on heteroassociative memory performance

David Casasent and Brian Telfer

Most associative memory work has concentrated on autoassociative memories (AAMs). These associative processors provide reduced noise and error correction in their output data. We will consider heteroassociative memories (HAMs), which are needed to provide decisions on the class of the input data and inferences for subsequent processing. We derive new equations for the storage capacity and noise performance of HAMs, emphasize how they differ from those derived for AAMs, suggest new performance measures to be used, and show how different recollection vector encodings can improve HAM performance.

I. Introduction

Much has been written[1-8] about associative memory storage capacity and the recollection and error correcting properties of autoassociative memories (AAMs). However, little attention[3,6,8] has been given to heteroassociative memories (HAMs). Our work discusses both types of associative memory and notes the differences in performance between each. Section II reviews pseudoinverse associative memory synthesis and recall, linear discriminant function analogies with pseudoinverse memories, and optical implementation of pseudoinverse memories and establishes our notation. In Sec. III, we introduce two performance measures for associative memories, and we consider new theoretical expressions for both pseudoinverse AAM and HAM storage capacity and noise performance. Quantitative data on noise reduction, SNR performance, and the probability of correct recognition for pseudoinverse AAMs and HAMs (with different recollection vectors) are then included in Sec. IV.

II. Associative Memory Synthesis and Recall

We review the types of associative memory and their synthesis (Sec. II.A). We then advance new analogies between pattern recognition, linear discriminant functions, and pseudoinverse and mean square error associative memories (Sec. II.B) and note new recollection vector encodings possible for HAM associative processors (Sec. II.C). Initial demonstration data (including associative recall of nonreference or training set images) is then included (Sec. II.D). Section II.E reviews the optical matrix–vector multiplier system that is attractive for pseudoinverse recall and discusses the importance of HAM capacity analysis and recollection vector choice in the optical implementation.

A. Notation and Pseudoinverse Associative Memories

We denote the key vectors of dimensionality N by $\mathbf{x}_k$ and the output recollection vectors of dimension K by $\mathbf{y}_k$. There are M key/recollection vector pairs (M is the storage capacity of the associative memory) and the purpose of the associative memory matrix $\mathbf{M}$ of dimensionality $K \times N$ is to recall the recollection vector $\mathbf{y}_k$ that is associated with the input key vector $\mathbf{x}_k$. Thus we desire $\mathbf{M}\mathbf{x}_k = \mathbf{y}_k$ for all $k = 1$ to M, where boldfaced upper case letters denote matrices and boldfaced lower case letters denote vectors. In terms of the key vector matrix $\mathbf{X}$ of size $N \times M$ (with the $\mathbf{x}_k$ as its columns) and the recollection vector matrix $\mathbf{Y}$ of size $K \times M$ (with the $\mathbf{y}_k$ as its columns), the associative memory must satisfy

$$\mathbf{MX} = \mathbf{Y}. \tag{1}$$

If $\mathbf{X}$ is square and nonsingular, we can write the solution as $\mathbf{M} = \mathbf{Y}\mathbf{X}^{-1}$. Generally $\mathbf{X}$ is not square, and thus the pseudoinverse solution

$$\mathbf{M} = \mathbf{Y}\mathbf{X}^{+} \tag{2}$$

is used, where

$$\mathbf{X}^{+} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T, \tag{3}$$

and $\mathbf{V} = \mathbf{X}^T\mathbf{X}$ is the vector inner product matrix. The

The authors are with Carnegie Mellon University, Department of Electrical & Computer Engineering, Center for Excellence in Optical Data Processing, Pittsburgh, Pennsylvania 15213

Received 15 December 1987.
0003-6935/89/020272-12$02.00/0.

solution in Eq. (2) with $\mathbf{X}^+$ given by Eq. (3) is useful only when the $\mathbf{x}_k$ are linearly independent, since then $\mathbf{V}$ is of full rank and can be inverted. In such cases, the pseudoinverse is an exact solution and is the minimum norm solution,[10] whose outputs $\mathbf{y}_k$ are least affected by input perturbations. If the $\mathbf{x}_k$ are orthonormal, then $\mathbf{X}^+ = \mathbf{X}^T$ and calculation of $\mathbf{M}$ is easy. Minimum squared error (MSE) solutions for the pseudoinverse exist when the key vectors are linearly dependent, but all such solutions are approximate. Such methods are not considered in this paper.

If the $\mathbf{x}_k$ are image domain vectors (i.e., lexicographically ordered images), and if $M \ll N$, often we will find that the $\mathbf{x}_k$ are linearly independent, or at least there is a reasonable assurance that this will occur. However, if the $\mathbf{x}_k$ are feature vectors or if M is large, the key vectors are generally linearly dependent. Thus there are many cases in which the assumption of linear independence is not valid. In such cases, advanced techniques such as those that produce an orthogonal basis from the key vectors (a vector inner product/Gram-Schmidt method[11] or an iterative eigenvector method[12]) can be used or the associative memory can be assembled using an iterative Widrow-Hoff method.[13] Several of these methods can be realized in real time. The use of the data matrix associative memory $\mathbf{M} = \mathbf{X}^T$ has also been suggested.[14] This memory has been shown to produce better nearest neighbor recollection than the popular Hopfield memory[4] for the cases of both binary[14] and gray scale[12] key vectors. When the input vector is not one of the original ones used to form $\mathbf{M}$, we desire the output y vector to be the one most closely associated with the input.

For AAMs, $\mathbf{Y} = \mathbf{X}$ and the output is an input key vector with reduced noise. For HAMs, the y output denotes the class of the input vector. In general, AAMs are employed for noise reduction and error correction and HAMs are used to provide decisions. Cascades of AAMs and HAMs have also been suggested and demonstrated[11] to achieve both goals. There are many possible algorithms and architectures that can be employed to synthesize and use associative memories. These include: forming $\mathbf{M}$ as a sum of vector outer products (for a pseudoinverse memory this requires orthonormal key vectors); performing the matrix–vector multiplications by vector inner products (this also requires orthonormal key vectors unless a nonlinearity is introduced in the central plane[15]); use of the iterative Widrow-Hoff technique[13] (this does not require linearly independent key vectors); the use of multiple memories with a voting scheme[16] to increase storage capacity (this is only achieved with an associated increase in the number of associative memories required); the use of an associative memory synthesized with higher-order correlations[7,9,17] (this increases storage capacity, but again at the cost of increased associative memory size, at least if the memory does not include nonlinear elements). Many associative memories are restricted to 0 and ±1 valued key vectors and memory matrix elements. Other associative memories use zero-valued diagonal matrix elements. Some associative memories use several matrix–vector iterations[4,7,17–19] to obtain the final result, whereas others employ only one pass through the system. Thus there are a large variety of associative memories. In our initial work here, we consider only the pseudoinverse associative memory. Our emphasis will be on HAMs and how their performance and storage capacity differ from those of AAMs. Our new results concern the performance of HAMs and how new recollection vector choices can improve capacity (increase M) and reduce memory size ($K \times N$). General expressions will be derived for associative memory performance with different conditions placed on the key vectors.

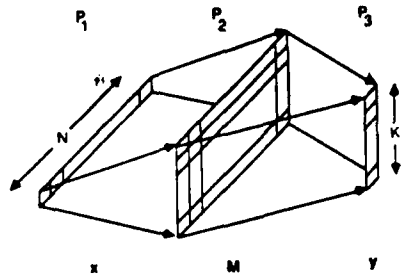


Fig. 1. Analog optical matrix–vector multiplier for pseudoinverse associative memory.

B. Pattern Recognition and Linear Discriminant Function Analogies

Conventional linear discriminant function (LDF) synthesis techniques are quite useful for HAM synthesis but are generally not employed. To show the analogy between the pseudoinverse solution in Eq. (2) and LDFs, we consider LDFs intended to recognize different input objects, discriminate them from other objects, and recognize different distorted versions of an object. This analogy will also prove useful in discussing different possible associative memory recollection vectors. The linear combination filter[20] is the sum of reference or training image vectors $\mathbf{x}_k$:

$$\mathbf{h} = \sum a_j \mathbf{x}_j = \mathbf{X}\mathbf{a}, \tag{4}$$

where the elements of the vector $\mathbf{a}$ denote the linear combination coefficients. Each filter $\mathbf{h}_k$ is designed to output a single element of the recollection vector $\mathbf{y}_k$. This is achieved by specifying each $\mathbf{h}_k$ as the solution of a matrix–vector equation $\mathbf{V}\mathbf{a}_k = \mathbf{u}_k$. The solution $\mathbf{a}_k = \mathbf{V}^{-1}\mathbf{u}_k$ specifies $\mathbf{h}_k$ in Eq. (4). When the filter solution is written as a row vector, the LDF solution is

$$\mathbf{h}_k^T = \mathbf{u}_k^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T = \mathbf{u}_k^T\mathbf{X}^+. \tag{5}$$

When K filters $\mathbf{h}_1$ to $\mathbf{h}_K$ with different output codings $\mathbf{u}_k$ for each object class are used, the result is analogous to the pseudoinverse associative memory with the rows of $\mathbf{M}$ being the different filter functions $\mathbf{h}_k$ and the rows of $\mathbf{Y}$ being the different $\mathbf{u}_k^T$ control vectors.

C. Recollection Vector Encoding

The LDF synthesis techniques suggest different recollection vector encodings. In the simplest case, each

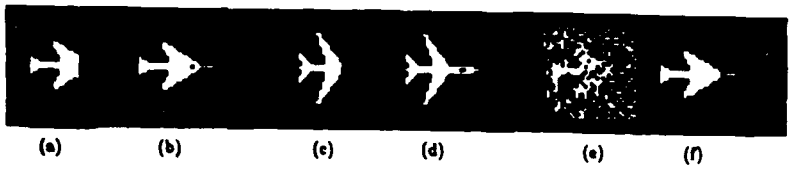


Fig. 2. Autoassociative memory image recollection: partial phantom input (a) and output obtained (b); partial DC10 input (c) and output obtained (d); noisy phantom input (e); and output obtained (f).

$\mathbf{u}_k$ is a unit vector (with a single one in a different location). In this case, $\mathbf{Y} = \mathbf{I}$ and the conventional[1] HAM results (with unit recollection vectors). However, consider the case when several key vectors $\mathbf{x}_k$ are assigned the same $\mathbf{y}_k$ recollection vector. Such cases arise when the $\mathbf{x}_k$ are members of the same class (e.g., different distorted versions of one object). The output vector y then denotes the class of the input object (independent of distortions etc.). The size $(K \times N)$ of M can now be reduced considerably, since the number of elements K in the recollection vector can be much less than M (the total number of $\mathbf{x}_k$ reference or training set key vectors). Another possible recollection vector encoding technique used in LDFs and suitable for associative memories is to use binary encoded $\mathbf{y}_k$ recollection vectors. In this case, a K element recollection vector and a $K \times N$ associative memory matrix can identify 2^K different key vectors or classes of key vectors (when more than one key vector is assigned the same recollection vector). For the case of M key/recollection vector pairs to be identified, K need only satisfy $2^K \geq M$. The value K required is thus significantly reduced from the $K = M$ required when unit recollection vectors are employed. We refer to these three different recollection vector encodings as (1) unit vectors, in which each $\mathbf{x}_k$ is assigned to a unique unit vector $\hat{\mathbf{y}}_k$ and thus $\mathbf{Y} = \mathbf{I}$; or (2) when several $\mathbf{x}_k$ are assigned to the same unit vector $\hat{\mathbf{y}}_k$; and (3) binary encoded recollection vectors (when the K element output vector can now accommodate 2^K different codes). Our theoretical and simulation analyses will employ these different recollection vector encodings and address the performance of the different HAMs that result.

D. Initial Associative Memory Examples

Figure 2 shows several examples of the AAM recall of partial and noisy input key vector image data. The left images show the input vector (in 2-D image format), and the right images show the output vector obtained (in the same 2-D image format). The AAMs used 36 key and recollection vectors, each of dimension $N = 32^2 = 1024$ (where the input images are 32×32 binary pixels). For the first AAM, the key vectors were 36 images of a Phantom jet at different yaw distortions (every 10°). We refer to this data base as PH-36. Figures 2(a) and (b) show the results when the input [Fig. 2(a)] is a Phantom at 0° with the first twelve columns of the Phantom removed. The second AAM was formed from 36 similarly distorted DC10 aircraft images. We refer to this data base as DC10-36. Figures 2(c) and (d) show the results from this AAM when the input is a DC10 at 0° with twelve columns of the input removed. The final example [Figs. 2(e) and (f)] shows the reconstruction when the input has binarized additive bipolar Gaussian noise with $\sigma = 0.6$. (This results in additional background noise as well as data dropout on the object.) These AAM reconstructions show the excellent noise reduction and error correction properties of AAMs.

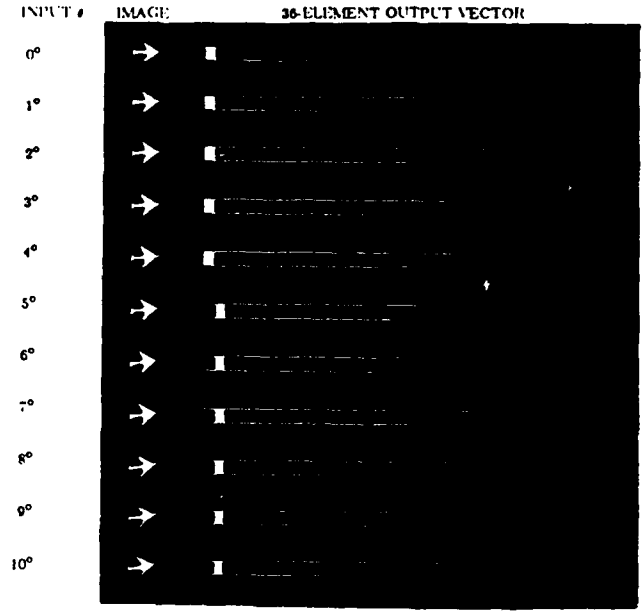


Fig. 3. Unit vector HAM 36-class demonstration. The input orientation is noted on the left, the object is shown in the center, and the output vector obtained is given on the right.

Figure 3 shows the results of a HAM object recognition associative memory employing unit recollection vectors. In this case, a 36×1024 element HAM was formed from the PH-36 database. The 36 unit recollection vectors have 36 elements in each. In this case, the output vector $[1,0,\ldots,0]^T$ denotes that the input is a Phantom at 0°, the output vector $[0,1,0,\ldots,0]^T$ denotes that the input is a Phantom at 10°, etc. The orientation of each image is indicated at the left of the figure, the input images are shown at the center of the figure, and the output vector obtained in each case is shown to the right. As seen, the HAM correctly denotes the reference or training set vector that is closest to the input test vector; i.e., the output vector obtained for the first five test inputs has a 1 in the first position, and the output vectors for the last five inputs have a 1 in the next position (indicating orientation within the 10° quantization levels set). The test inputs contain nontraining set image data, and thus the results indicate distortion-invariant performance of the associa-

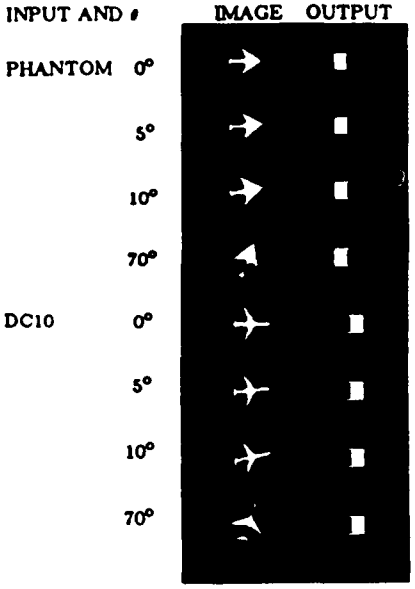


Fig. 4. Binary vector HAM two-class demonstration. The recollection vectors $[1,0]^T$ and $[0,1]^T$ denote a Phantom and DC-10, respectively. The input orientation is noted on the left, the object is shown in the center, and the output vector obtained is shown to the right.

tive processor with simple hyperplane discriminant surfaces used (rather than the piecewise linear discriminant surfaces that would result from a nearest neighbor classifier).

Figure 4 shows the results of a reduced class 2×1024 HAM formed from the PH-18 and DC10-18 set of key vectors with only a $K = 2$ element recollection vector used. For this HAM, the recollection vector $[1,0]^T$ was assigned to all eighteen Phantom key vectors, and the recollection vector $[0,1]^T$ was assigned to all eighteen DC-10 key vectors. The data in Fig. 4 show the ability of the memory to classify correctly training and nontraining set input vector imagery. As noted earlier, the memory size and recollection vector dimensionality are significantly reduced with this encoding. These data represent one of the first examples of the use of an associative processor for multiclass distortion-invariant data recognition.

E. Optical Implementation of Pseudoinverse Memories

We propose implementing pseudoinverse memory recall on the standard optical analog matrix–vector multiplier shown in Fig. 1. The optical system is attractive for its high-speed parallel computing power. Analog operation is desirable for speed and simplicity, and this is possible since simulations[11] have shown that the pseudoinverse memory operates well in low dynamic range systems. The system operates as follows. The P_1 input plane contains N point modulators with light outputs proportional to $\mathbf{x}$. Each element of $\mathbf{x}$ uniformly illuminates one column of the memory $\mathbf{M}$, a transmittance array of $K \times N$ elements at P_2, and the light leaving P_2 is integrated horizontally onto K detectors at P_3. The detector output is thus the desired matrix–vector product $\mathbf{y} = \mathbf{M}\mathbf{x}$. The matrix $\mathbf{M}$ will be bipolar. We note that a variety of techniques have been developed for representing bipolar data.[21-24]

As mentioned in Secs. II. C and II. D, certain choices of recollection vectors result in memory matrices much smaller than the conventional $\mathbf{Y} = \mathbf{I}$ memory matrix. Such choices simplify the optical architecture (Fig. 1) by reducing the size of the spatial light modulator (SLM) needed at P_2 and the number of parallel detector outputs required at P_3. On the other hand, different recollection vector choices affect memory capacity and recall accuracy. Thus, we consider the effects of such reduced size recollection vectors on the system's performance rather than simply choosing recollection vectors that minimize the optical architecture's size. These effects on memory capacity and recall accuracy are discussed in Sec. III and IV.

III. Noise Performance and Storage Capacity: Theory

After introducing the notation to be used (Sec. III.A), we summarize prior relevant work on associative memory capacity and noise performance and advance four relevant theorems (Sec. III.B). We then present new equations for evaluating the performance measure (Sec. III.D), and we introduce a new performance measure and expressions for it (Sec. III.E).

A. Notation

The storage capacity M and noise performance of associative memories must be considered together. Most work has addressed autoassociative memories or specific neural associative memories, most notably the Hopfield memory.[4,18] Our concern lies with general pseudoinverse associative memories without specific conditions on the key vectors and with HAMs rather than only AAMs. The noisy input key vectors are described as $\mathbf{x} = \mathbf{x}_k + \mathbf{n}$, where $\mathbf{x}_k$ is the key vector and $\mathbf{n}$ is the noise vector of zero-mean Gaussian noise with a covariance matrix $\sum = \sigma_i^2\mathbf{I}$. The variances of the input and output noise are denoted by σ_i^2 and σ_o^2 where the variance of a random variable x is $\sigma_x^2 = E\{x^2\} - E\{x\}^2$ and for zero-mean data, $\sigma_x^2 = E\{x^2\}$. Both the input and output noise are zero-mean, since the associative memory matrix operator $\mathbf{M}$ is linear (if no output thresholding is employed). We use subscripts to denote specific vectors in a set and superscripts to denote the elements of a vector. In this notation, $\sigma_i^2 = E\{(n^i)^2\}$ and $\sigma_o^2 = E\{(y^i - y_k^i)^2\}$, where $\mathbf{y} = \mathbf{M}\mathbf{x}$ and where the recollection vector $\mathbf{y}_k$ corresponds to the key vector $\mathbf{x}_k$ and the expectation is over the elements of the vectors.

B. Prior Results

The typical associative memory performance measure used has been σ_o^2/σ_i^2, where small values of this parameter indicate good performance. Kohonen[1] proved for AAMs that

$$\sigma_o^2/\sigma_i^2 \sim M/N \tag{6}$$

and reasoned that the result for HAMs would be about the same. Other work[3] showed this to be incorrect. The documentation of this work is very terse, and thus

it merits more details. We provide these in the Appendices. These results are now highlighted. Monte Carlo simulations were performed for the AAM case,[3] with the key vectors chosen from a uniform distribution between −1 and +1 and with the key vectors required to be linearly independent. (This we found to be a requirement for the cases when $M \leq N$, although it is not noted in the original work.[3]) In other tests[3] of these associative memories, the input test vectors were formed by adding a zero-mean random variable (with uniform distribution over −1 to +1) to each element of one of the random key vectors. For each associative memory matrix, ten input vectors (each of length N) were tested using one key vector with ten different added realizations of noise with the same level σ_i. Different M/N ratios were tested by fixing $N = 50$ and by varying M (with ten different input vectors used to test each memory matrix **M**). For the case of an HAM, each element of the N-element recollection vector was chosen from a uniform distribution between −1 and +1. (These recollection vectors had more than one 1 and are thus not unit vectors.)

We define the signal power of a vector to be $E\{(x^i)^2\} - E\{x^i\}^2$. This definition subtracts the vector's mean from all elements and then calculates the average squared element value for each vector. Since the key and recollection vectors were chosen in the same manner in these earlier tests,[3] their signal powers are equal, and σ_o^2/σ_i^2 is equivalent to the input-to-output SNR. We now state four theorems.[3] Proofs of each are provided in the Appendices.

Theorem 1: For any matrix recollection $\mathbf{y} = \mathbf{Mx}$, we find $\sigma_o^2/\sigma_i^2 = NE\{m_{ij}^2\}$, where m_{ij} is an element of **M** and the expectation is over all elements of **M**.

Theorem 2: For an AAM with linearly independent key vectors, we find $E\{m_{ij}^2\} = M/N^2$.

Theorem 3: For AAMs with linearly independent key vectors, combining Theorems 1 and 2, we immediately find

$$\sigma_o^2/\sigma_i^2 = M/N. \tag{7}$$

Theorem 4: For HAMs, we find

$$\sigma_o^2/\sigma_i^2 = E\{y_{ij}^2\}E\{\mathrm{Tr}(\mathbf{V}^{-1})\}, \tag{8}$$

where y_{ij} is an element of **Y**, $\mathbf{V} = \mathbf{X}^T\mathbf{X}$, and the trace (Tr) is the sum of the diagonal elements of the matrix noted in parentheses following this operator. The first expected value operator is taken over all elements of **Y**, and both expectation operators are taken over the entire ensemble of possible key and recollection vectors.

C. Discussion of Prior Results

Theorem 1 is useful, since it applies for any matrix, with no key or recollection vector assumptions. We will use it in Sec. III.D to develop more general and more easily evaluated expressions of associative memory performance. The result in Theorem 3 agrees with that of Kohonen,[1] who obtained his result by quite different techniques. This result shows and quantifies the classic AAM result for linearly independent key vectors, i.e., that an AAM always reduces the input noise (or in the worst case, when $M = N$, the input noise is not increased). This also shows that the noise improvement for an AAM is better as M/N decreases (i.e., as fewer vector pairs M are stored or when larger dimensionality-N key vectors are used). For an AAM design, when the expected amount of input noise σ_i^2 is specified, this expression shows that M/N determines the output noise σ_o^2 that one will have to contend with. Theorem 4 is quite significant, since it shows that the amount of noise reduction in an HAM depends on the key vectors [this occurs through the $\mathrm{Tr}(\mathbf{V}^{-1})$ term] and that it also depends on the recollection vector choices made (this occurs through the y_{ij} term) and that its performance does not depend on simply M and N as was the case for an AAM. In deriving Theorem 4, it was assumed that all recollection vectors had the same energy, but that the energy of the recollection vectors need not equal that of the key vectors.

The ensemble averages in Eq. (8) make evaluation of the performance measure impossible, except by Monte Carlo techniques. In these Monte Carlo techniques, one computes the performance measures σ_o^2/σ_i^2 by averaging over a number of different associative memories (with different key and recollection vector pairs). These results[3] are not a good estimate of the performance for specific associative memories. These prior results[3] used recollection vectors that were random (containing more than a single 1) and used recollection vectors whose energy was equal to that of the key vectors. Thus these results are appropriate for an AAM but not for the conventional types of HAM. In addition, if the dimensionality of the key and recollection vectors are different, the prior results[3] are not of use. Thus other σ_o^2/σ_i^2 expressions are desirable. These are provided in Section III.D.

D. More General σ_o^2/σ_i^2 Expressions

Alternate σ_o^2/σ_i^2 expressions can be obtained in the case of unit recollection vectors $\hat{\mathbf{Y}} = c\mathbf{I}$, where c is a constant. In this case of HAMs, we find

$$\sigma_o^2/\sigma_i^2 = (c^2/K)\mathrm{Tr}[\mathbf{V}^{-1}]. \tag{9}$$

The second case in which an equation without all expected value operators is possible occurs for the case of orthogonal key vectors. In this case of HAMs with orthogonal key vectors,

$$\sigma_o^2/\sigma_i^2 = E\{y_{ij}^2\}\mathrm{Tr}[\mathbf{V}^{-1}], \tag{10}$$

where the expectation operator is the average over all squared elements of $\hat{\mathbf{Y}}$. Since c^2/K in Eq. (9) equals $E\{y_{ij}^2\}$ for $\hat{\mathbf{Y}} = c\mathbf{I}$, Eq. (10) is equivalent to Eq. (9). Thus, in terms of performance, assuming unit recollection vectors are analogous to using orthogonal key vectors. This is a noteworthy new result, since one might feel that orthogonal key vectors would yield better performance. This result follows from linear algebra, since **V** (and $\mathbf{V}^{-1}$) are then diagonal if the key vectors are orthogonal, thus yielding only the trace elements of the matrix.

For cases when no conditions on the recollection vectors $\mathbf{y}_k$ (such as unit recollection vectors) are made and similarly when no conditions on the $\mathbf{x}_k$ key vectors are made, Theorem 1 can be used. An alternate σ_o^2/σ_i^2 expression can then be found by substituting Eqs. (D3) and (D6) in the Appendices into Theorem 1 to obtain

$$\sigma_o^2/\sigma_i^2 = (1/K) \sum_i \sum_m \sum_k v_{mk}^{-1} y_{im} y_{ik}, \tag{11}$$

where v_{mk}^{-1} is the mkth element of $\mathbf{V}^{-1}$. Equation (11) is equivalent to Theorem 1. However, calculations using Eq. (11) are preferred since it provides the result without the need to first explicitly compute $\mathbf{M}$.

In our quantitative test data, we will use Eqs. (7), (9), and (11) for different cases. Equation (7) applies for AAMs with linearly independent key vectors, and Eq. (9) applies for HAMs with linearly independent key vectors and with unit recollection vectors. Equation (10) applies for orthogonal key vectors, and Eq. (11) has no conditions on the recollection vectors or the key vectors.

E. Preferred SNR Associative Memory Performance Measure

With one exception,[8] all prior theoretical studies[1,3] of pseudoinverse associative memory noise performance have used the σ_o^2/σ_i^2 performance measure. Other work on associative memory capacity either does not consider HAMs, yields bounds (not exact expressions), or does not consider noise. The σ_o^2/σ_i^2 performance measure is valid for AAMs, but not for HAMs, since its resultant value can be reduced (improved) artifically by merely reducing the energy of the recollection vectors (i.e., by using unit rather than binary-encoded recollection vectors). Our σ_o^2/σ_i^2 data verify that unit recollection vectors perform better than binary encoded ones. To see the problem with the σ_o^2/σ_i^2 measure, consider Theorem 1 for the case of an HAM. If we scale each $\mathbf{x}_k$ by a constant c_x and each $\mathbf{y}_k$ by a constant c_y, the new associative memory matrix is $\mathbf{M}' = (c_y/c_x)\mathbf{M}$, where $\mathbf{M}$ is the original associative memory matrix. The new expected value (denoted by an apostrophe) is related to the expected value for the original matrix (denoted by no apostrophe) by $E\{m_{ij}^2\}' = (c_y^2/c_x^2)E\{m_{ij}^2\}$. The new and old performance ratios are thus related by $(\sigma_o^2/\sigma_i^2)' = (c_y^2/c_x^2)\sigma_o^2/\sigma_i^2$. From this, we see that increasing c_x/c_y results in an improved new σ_o^2/σ_i^2 ratio. However, this improvement is artificial. We note that this issue does not arise for the case of an AAM (since for this matrix, its recollection and key vectors are the same and thus have the same energy and scaling factors). These remarks also do not apply to earlier results,[3] where equal energy key and recollection vectors were used in the Monte Carlo data obtained. This σ_o^2/σ_i^2 performance could be applied to an HAM with $\mathbf{Y} = \mathbf{I}$ (or to binary-encoded recollection vectors or to recollection vectors whose dimensionality $K < N$) by appropriately scaling the recollection vectors, so that their energy and that of the key vectors are the same. In general, with arbitrary key vectors and unit or other possible recollection vector encoding schemes, the need exists for a different performance measure.

The performance measure we introduce is the output-to-input SNR, $\mathrm{SNR}_o/\mathrm{SNR}_i$. The larger this ratio, the better the performance. For equal key and recollection vector energies, this measure and σ_o^2/σ_i^2 are reciprocals. We define the signal powers as the expected value of the square of the elements minus the square of the expected value of the elements; i.e., we subtract off the average or bias energy from our calculations of signal energy. Thus the signal energies we use are

$$s_i^2 = E\{|x_k^i|^2\} - E\{x_k^i\}^2, \tag{12a}$$

$$s_o^2 = E\{|y_k^i|^2\} - E\{y_k^i\}^2, \tag{12b}$$

where the energy values are averages over all elements i of all vectors k. The resultant SNR performance ratio is then

$$\frac{\mathrm{SNR}_o}{\mathrm{SNR}_i} = \frac{s_o^2 \sigma_i^2}{s_i^2 \sigma_o^2}. \tag{13}$$

For AAMs (with $s_o^2 = s_i^2$), Eq. (13) reduces to N/M (from Theorem 3), which is the reciprocal of Theorem 3.

Our concern lies with HAMs. For HAMs with unrestricted key vectors, we combine Eqs. (11) and (13) to obtain

$$\frac{\mathrm{SNR}_o}{\mathrm{SNR}_i} = \frac{s_o^2 K}{s_i^2 \sum_i \sum_m \sum_k v_{mk}^{-1} y_{im} y_{ik}}, \tag{14}$$

For HAMs, with $\mathbf{Y} = c\mathbf{I}$ (or for the case of orthogonal key vectors), we combine Eqs. (10) and (13) to obtain

$$\frac{\mathrm{SNR}_o}{\mathrm{SNR}_i} = \frac{s_o^2}{s_i^2 E\{y_{ij}^2\} \mathrm{Tr}\{\mathbf{V}^{-1}\}}. \tag{15}$$

For zero-mean key and recollection vectors, $s_o^2 = E\{y_{ij}^2\}$ and $s_i^2 = E\{x_{ij}^2\} = (1/M)\mathrm{Tr}\{\mathbf{V}\}$. Under these assumptions, HAMs with $\mathbf{Y} = c\mathbf{I}$ (or HAMs with orthogonal key vectors) yield

$$\frac{\mathrm{SNR}_o}{\mathrm{SNR}_i} = \frac{M}{\mathrm{Tr}[V]\mathrm{Tr}[V^{-1}]} = \frac{1}{M}, \tag{16}$$

where the last equality holds for orthonormal key vectors, since $\mathbf{V} = \mathbf{X}^T\mathbf{X} = \mathbf{I}$ and $\mathrm{Tr}(\mathbf{V}) = \mathrm{Tr}(\mathbf{V}^{-1}) = M$ for this case. We will employ the different performance measures noted in Eqs. (13)–(15) in our quantitative comparison tests of performance in Sec. IV.

IV. Quantitative Test Data

Section IV.A details the data bases and associative memories used and our test procedure. Our σ_o^2/σ_i^2 and SNR performance measure test results follow in Secs. IV.B and IV.C. Initial larger class test remarks (Sec. IV.D), probability of correct recognition expressions (Sec. IV.E), and a general design procedure (Sec. IV.F) are then provided.

A. Test Procedure

The data base used for most associative memory tests to verify and quantify the theory advanced in Sec. III consisted of the $M = 36$ set of key vectors from the PH-18 and DC10-18 data bases. Our tests on larger data bases with $M = 72$ used the PH-36 and DC10-36 data bases noted earlier. Three different associative memories were synthesized. We formulated an AAM (with $\mathbf{Y} = \mathbf{X}$), a HAM-1 (with $\mathbf{Y} = \mathbf{I}$, with unit recollections), and a HAM-2 reduced size binary encoded associative memory (with two element recollection vectors $[1,0]^T$ and $[0,1]^T$). All initial associative memories were synthesized from both Phantom and DC-10 key vectors (eighteen images of each class). For the AAM, the conventional matrix would be 1024×1024, with the output vector being 1024×1. However, the rank of the matrix is only 36, and the input and output vectors can be described in terms of 36 basis functions (determined by the Gram-Schmidt technique[11]). In our AAM tests, we used 1024×1 image domain input vectors and a 36×1024 element matrix with matrix rows being the basis vectors. The 36-element output vector thus specifies the linear combinations of the 1024×1 basis functions that yield the final image domain output vector. This basis function matrix is of significantly reduced size and allows efficient testing of the AAM. The purpose of this AAM was to produce a noise free output image (as in Fig. 2). For HAM-1, the key vectors are 1024×1, the matrix is 36×1024, and the recollection vectors are 36-element unit vectors. The output here denotes the class and the orientation of the input vector data. For HAM-2, the matrix is 2×1024, and the recollection vector is two elements long, with the outputs $[1,0]^T$ and $[0,1]^T$ denoting the object class (a Phantom or a DC-10) independent of the orientation of the input object.

For noise tests, the 0° Phantom image was used, and zero-mean Gaussian noise with five different standard deviations σ_i was added to the 0° Phantom image. For each σ_i noise level, ten different input images (test vectors) with the same σ_i level, but with different seeds (i.e., different statistical realizations), were used. In all noise tests, the noisy input images were not rebinarized (to allow better comparison between theory and tests). If binary light modulators were employed in the processor, they would rebinarize the input key vectors. We feel that the performance obtained with gray level noise is comparable to what would result with binarized noise.

For all memories $\mathbf{X}^+$ was calculated using the IMSL generalized inverse subroutine.[25] All 36 or 72 key vectors were found to be linearly independent. This was verified from a calculation of the condition number of the VIP matrix ($\lambda_{max}/\lambda_{min} = 183$ for the $M = 36$ case), which showed that V was of full rank.

B. Associative Memory Test Results Using the σ_o^2/σ_i^2 Measure

Our initial test results are summarized in Table I. Each entry in this table is the average of ten realizations of noise with the standard deviation listed. The performance measure tabulated is σ_o^2/σ_i^2 for the AAM and two HAMs constructed. The average of the measured σ_o^2/σ_i^2 values for all 50 noise image tests for each associative memory is given in the bottom of the table. The theoretical value for the AAM is calculated as M/N from Eq. (7), and it agrees quite well (within 3%) with the measured average. For both HAMs, theory and experiment also agreed quite well (within 1.5 and 11%). The theoretical values for HAM-1 (with unit recollection vectors) were calculated from the trace of $\mathbf{V}^{-1}$ in Eq. (9) with $c = 1$ and $K = M = 36$. For HAM-2 with only $K = 2$ output elements, we calculated the theoretical value using Eq. (11). Several initial obvious remarks are in order. First, we note general good agreement between theory and tests. Second, we note that HAM-1 performance is 50% better than that of the AAM. (The lower σ_o^2/σ_i^2 performance measures indicate better performance.) One would not expect HAM performance to be better than that of an AAM. We investigate this further in Sec. IV.C.

Our final comments concern the performance of the two HAMs. The second HAM (with only two distinct recollection vectors and two recollection vector elements) performed worse. This is unfortunate but will be overcome in Sec. IV.C and IV.D. This occurs since this matrix is 2×1024 with its first row being a sum of the first eighteen rows of the first HAM and its second row being a sum of the second eighteen rows of the first HAM. Recall that the size of the first HAM-1 is 36×1024. In this case, summing the rows of $\mathbf{M}$ increases $E\{m_{ij}^2\}$ and causes an increase in σ_o^2/σ_i^2 (and thus poorer performance). In general, summing the rows of the first HAM will not always increase $E\{m_{ij}^2\}$, since the elements of $\mathbf{M}$ are bipolar. Here an increase occurred, because the key vectors corresponding to the added rows are members of the same class (rotated yaw views of the same aircraft) and are thus similar, causing the added rows to be similar. We will discuss these results further after our SNR performance measure data have been presented in Sec. IV.C.

Table I. σ_o^2/σ_i^2 for AAM and HAM

σ_i	$\frac{\sigma_o^2}{\sigma_i^2}$		
	AAM	HAM-1 Y=I	HAM-2 [1,0]T, [0,1]T outputs
0.2	0.0352	0.0220	0.0949
0.3	0.0359	0.0218	0.153
0.4	0.0400	0.0253	0.0949
0.5	0.0323	0.0180	0.201
0.6	0.0387	0.0236	0.0655
average	0.0364	0.0221	0.122
theory	0.0352	0.0218	0.136

C. Associative Memory Test Results Using the SNR_o/SNR_i Measure

We now test and compare our three associative memories using our SNR performance measure. Our results are shown in Table II. Larger values for this performance measure indicate better performance. In each case, the data presented are the average of fifty runs for five different noise σ_i values, with the measured data obtained from image domain tests. These measured data are then compared to the associated theoretical equations. The AAM results are the reciprocals of those given in Table I. For HAM-1 (with unit recollection vectors), s_o^2/s_i^2 is small and for HAM-2 (with $[1,0]^T$ or $[0,1]^T$ recollection vectors) this ratio is large (since HAM-1 has more zeros in each recollection vector). Thus the SNR performance of HAM-2 is better than for HAM-1 (although its σ_o^2/σ_i^2 performance was worse). Equation (13) and Table I were used for all theoretical calculations in Table II. From these specific tests, we find AAM noise performance to be better than HAM noise performance (as one would expect) and that different HAMs (such as those with $K = 2$ output elements, the number of general classes of the data) are preferred to the conventional HAMs (with $\mathbf{Y} = \mathbf{I}$ unit recollection vectors with $K = M = 36$ elements and 36 output unit vectors). This represents a new result. These quantitative results in Table II are not necessarily general trends but are data dependent as we now discuss.

The performance of an AAM depends solely on the M and N values. HAM performance depends on $\mathbf{V}^{-1}$ with HAM-1 performance depending only on the diagonal elements of $\mathbf{V}^{-1}$ (because DC10s are slightly larger than Phantoms, the diagonal elements are not the same) and with HAM-2 performance depending on all elements of $\mathbf{V}^{-1}$. Since HAM performance depends on the key vectors used, no general conclusion on AAM vs HAM performance is possible. However, in our example, the HAM with a new (binary) recollection vector coding consistently performed better than the HAM with conventional unit recollection vectors. Our theory in Sec. III predicted this (for the SNR ratio performance measure). The presence of the elements of Y (recollection vectors) in our equations in Sec. III confirms this theoretically, and our test data in Table II quantify it. As discussed, s_o^2/s_i^2 is better for HAM-2, which is the reason our new HAM-2 outperforms HAM-1.

Table II. SNR_o/SNR_i for AAM and HAM

	SNR_o/SNR_i		
	AAM	HAM-1 Y=I	HAM-2 $[1,0]^T$, $[0,1]^T$ outputs
average	27.47	9.14	15.33
theory	28.41	9.26	13.75

D. Large Class Problems

The concern in associative processors should be large class problems (M large). We now briefly consider how AAM and HAM performance varies with M/N. We expect the performance to decrease as M/N increases. From Eq. (7), we expect AAM performance to reduce linearly as M increases. For HAMs, the performance variation with M will depend on the specific data. Table III shows initial results. Equations (7), (15), and (14) were used for the three associative memories, respectively. The second database used thirty-six images of each aircraft at 10° yaw increments and thus represents a larger $M = 72$ class problem. AAM performance is seen to be linear with M and thus reduces by a factor of 2 as shown. The reduction for the HAMs is data dependent. From these data, we clearly see that HAM performance does not degrade as fast as AAM performance and that at $M = 72$, the performance of HAM-2 and the AAM are approaching each other. Again, this result is not a general trend that we can always be assured of (since HAM performance is data dependent). However, this lends further justification for attention to HAM storage capacity and noise performance and to different recollection vector encoding schemes.

E. Probability of Correct Recognition for Different Recollection Vectors

We next consider how the σ_o level of the output noise affects the probability of correct recognition of the output vector y. For this case, we add zero-mean Gaussian noise equally distributed to all bit elements of the recollection vector. We consider a sixteen-class ($C = 16$) problem ($M > C$) with the recollection vector required to distinguish sixteen different classes of key vector. We consider four different recollection vector encodings. These are detailed below, with P_C (the probability of correct recognition) derived for each case.

The first encoding is a binary encoding with each element of the output vector thresholded at 0.5. The probability of a single bit error for zero-mean Gaussian noise of standard deviation σ_o is

$$p = 0.5 - \mathrm{erf}(0.5/\sigma_o), \tag{17}$$

where erf denotes the error function. For this encod-

Table III. Associative Memory SNR_o/SNR_i Performance as M Increases

DATA-BASE	M	SNR_o/SNR_i		
		AAM	HAM-1 Y=I	HAM-2 $[1,0]^T$, $[0,1]^T$ outputs
PH-18/ DC10-18	36	28.41	9.26	13.75
PH-36/ DC10-36	72	14.22	6.56	11.84

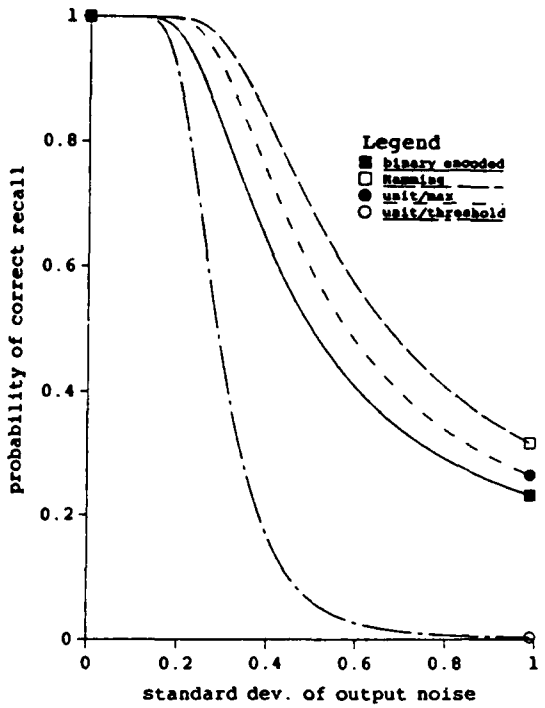


Fig. 5. Probability of correct recognition P_C vs output noise standard deviation σ_0 for four different recollection vector encoding schemes.

ing, P_C for an error in any of the K output vector digits is

$$P_C = (1-p)^K = [0.5 + \text{erf}(0.5/\sigma_o)]^K. \tag{18}$$

The second encoding considered uses unit recollection vectors, with the output threshold set at 0.5. For this case, P_C is also given by Eq. (18). However, $K = C$ digits are now required (whereas for binary encoded vectors, we require K digits where $2^K \geq C$). For binary encoding, the optimum threshold to maximize P_C is 0.5 (assuming all classes are equally likely to occur), since the probabilities of zero or one recollection vector elements are equal. However, for unit recollection vectors, the probability of a one occurring is only $1/K$, and the optimum threshold is larger than 0.5. The optimum threshold in this case varies with σ_o. In many cases, one has an indication of the σ value to be expected. Even with the optimum threshold, unit vectors with thresholding result in poorer performance than binary encoded vectors, when the output noise is the same. This occurs since the unit vectors have higher dimensionality.

To provide better performance with unit vector encoding, we consider (as our third type of recollection vector) unit vector encoding with the largest element of the recollection vector detected and set to one. As we shall see, this provides significant improvement in P_C for unit vectors. The output vector in this case will be correct if the largest element is in the correct position. To find P_C for this encoding and thresholding, we let T be the value of the output element that corresponds to the one in the correct recollection vector. We represent the probability that the level T occurs as $P(T)$ and the probability that all the other $K-1$ elements are less than T by $[0.5 + \text{erf}[T/\sigma_o]]^{K-1}$. Strictly speaking, we cannot evaluate $P(T)$ because $P(T) = 0$; thus we will evaluate $P(T-\delta < x < T+\delta)$. We can evaluate this quantity since $T = G(1,\sigma_o)$ for the element that should be one with the other output elements being given by $G(0,\sigma_o)$, where $G(m,\sigma)$ denotes a Gaussian random variable with mean m and standard deviation σ. P_C is then the integral (over all possible T values) of the probability that T can occur times the probability that all the other $K-1$ elements are less than T, i.e.,

$$P_C = \int P(T)[0.5 + \text{erf}[T/\sigma_o]]^{K-1} dT. \tag{19}$$

This equation cannot be evaluated in closed form, even when σ_o is specified, and thus in our calculations it is evaluated numerically.

The final case we consider is an HAM with error correcting binary recollection vectors. This associative memory concept[26] uses the Hamming code in which the n recollection vector bits encode k bits of data (2^k object classes). For a sixteen-class problem, we use $(n,k) = (7,4)$, i.e., a $K = 7$ element recollection vector rather than only a four-element one. This code can correct one output bit error, and thus for this case

$$P_C = (1-p)^K + Kp(1-p)^{K-1}, \tag{20}$$

where p is given by Eq. (17) and where we assume output elements are thresholded at 0.5.

Figure 5 shows P_C vs σ_o graphically for the four recollection vector cases noted using Eqs. (17)–(20). The data clearly demonstrate that unit vectors with a 0.5 threshold performed worst of all and that maximum element detection is necessary to make the performance of unit recollection vectors comparable to other output encoding schemes. We note that in this case, unit vector recollections with maximum element detection give better P_C performance than binary encoded recollections. However, the unit vectors have a higher dimensionality and require a larger matrix. In addition, maximum element detection requires a more complicated postprocessor. As expected, error correction binary encoding offers the best performance of all. (This is achieved at the expense of a more complex decoding output system, which can be achieved with a small associative memory as detailed elsewhere.[26]

F. Designs Combining σ_o^2/σ_i^2 and P_C Performance Criteria

Our prior work (involving the σ_o^2/σ_i^2 performance measure) has shown and quantified the noise reduction that an HAM achieves. We have also shown (through our P_C performance measure) the tolerance to the output noise that different recollection vector encodings can provide. We now combine these two performance measures to include the noise reduction achieved both by the memory itself and by our recollection vector choice. As a specific example, we consider an HAM with $M = 36$ designed to recognize and discriminate thirty-six key vectors in the PH-18 and DC10-18 databases. From Fig. 5, we find that for nearly perfect performance ($P_C \simeq 1.0$), we can tolerate

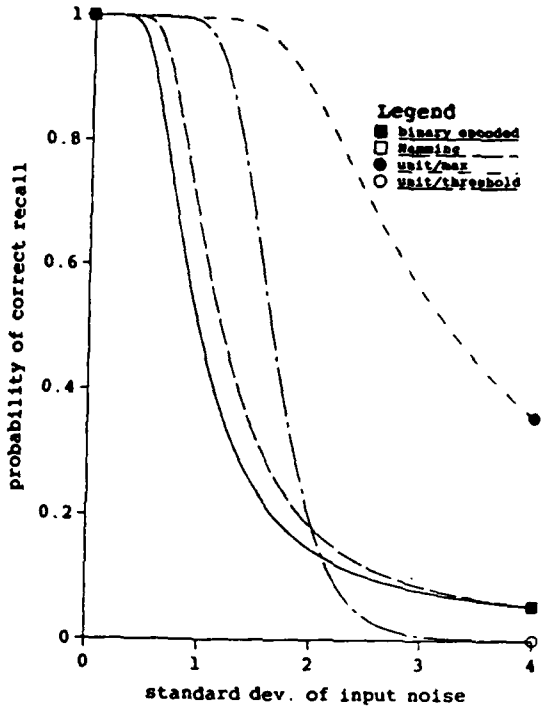


Fig. 6. Probability of correct recognition P_C vs input noise standard deviation σ_i for four different recollection vector encoding schemes.

an output noise standard deviation of $\sigma_o = 0.2$ in the recollection vector. From Table I, we find that the HAM-1 *with unit vector encoding* provides a noise reduction of $\sigma_o^2/\sigma_i^2 = 0.0218$. Thus this associative memory can tolerate input noise with a variance of

$$\sigma_i^2 = \sigma_o^2/0.0218 = (0.2)^2/0.0218 \leq 1.83 \qquad (21)$$

and still achieve nearly perfect ($P_C = 1$) performance. We now consider P_C vs σ_i for the different recollection vector encodings.

Using the PH-18 and DC10-18 databases as key vectors, we computed the total recall accuracy P_C as a function of σ_i for our four types of recollection vector. To have a relatively large number of classes, we chose each key vector to be a separate class. This resulted in thirty-six classes, with the unit recollection vectors having thirty-six elements. For the binary encoded vectors, we used six-element recollection vectors. The particular vectors were chosen so that the optimum threshold would be 0.5. For the Hamming vectors, a (7,4) code could only represent sixteen classes, so we chose the next smallest code, a (15,11) code that can represent 2^{11} classes. Since this has many unused states, the particular vectors used were chosen so that the optimum threshold would be 0.5. Although a (15,11) Hamming code is nominally 15 bits long, only 10 bits were nonzero for the thirty-six class case, and the Hamming vectors thus contained 10 bits.

The total recall accuracy was computed from Eqs. (17)–(20). These equations assume identical noise power for each output element, and to use the equations we assume that the noise in each element equals the average output noise power σ_o^2. This is an approximation since the noise power does differ for each output element. For our data, this approximation differed at most by 2.5% from a more involved exact analysis taking into account the different noise powers for each output element.

The results are shown in Fig. 6. Our estimate of nearly perfect recall for $\sigma_i^2 \leq 1.83$ ($\sigma_i \leq 1.35$) with unit vector recollections and maximum element detection is verified by the graph (with $P_C = 1$ for $\sigma_i = 1.35$). The unit vector recollections, with either type of detection, give the best performance. The primary reason for this is the unit vector HAM's large σ_o^2/σ_i^2 advantage (see Table I). The fact that the binary encoded and Hamming vectors have unused states also contributes to their relatively poor showing but is a secondary factor. As expected, the Hamming vector recollections perform better than the binary encoded vectors. Since σ_o^2/σ_i^2 is data dependent, we hesitate to reach general conclusions about which recollection vector will have the best total recall accuracy. However, all four types perform well. Choosing one type of recollection vector over another will largely depend on the dimensionality of the recollection vectors. This makes the binary encoded and Hamming vectors preferred to unit vectors for large class problems.

V. Summary and Conclusion

We have detailed several proofs of the noise performance and storage capacity M for HAMs using the noise variance reduction ratio σ_o^2/σ_i^2 as a performance measure. New σ_o^2/σ_i^2 expressions for HAMs were derived that did not require Monte Carlo techniques for evaluation. Quantitative data were obtained on an AAM and two HAMs with different recollection vector encodings on thirty-six-class distorted images of two aircraft. From these data, we give evidence that the performance of an HAM (using the σ_o^2/σ_i^2 performance measure) may actually exceed the performance of an AAM. A new and preferred SNR ratio performance measure was advanced, and new equations for this performance measure were derived for different HAMs. Quantitative test data obtained with this measure showed that the type of recollection vector encoding selected has a significant effect on performance. We found (from this SNR standpoint) that our reduced class HAM (which denotes each object class by a single recollection vector, independent of distortions in the key vectors) performed better than the more conventional unit recollection vector HAM.

We derived expressions for the probability of correct recognition for four different recollection vector encodings (unit vectors with 0.5 thresholding and with maximum element detection, binary, and Hamming error-correction binary codings) as a function of σ_o. We found (from this standpoint) that unit vectors with thresholding at 0.5 gave the worst performance, binary encoded recollections gave better results, and unit vectors with maximum element detection and Hamming vectors gave the best results.

We then considered total recall accuracy (as a function of σ_i) and obtained our final conclusions. We

found that (for a small number of classes), unit vectors with maximum element detection are preferred because they give excellent performance and because it is relatively simple to detect the largest element with only a few output elements. For a large number of classes, the binary encoded and Hamming recollections become preferable. Even if the unit vector recollections offer better recall, their dimensionality (and that of the matrix) becomes so large that their use becomes impractical. Considering binary encoded vs Hamming recollections, binary encoded vectors are smaller and will allow a smaller memory matrix, whereas Hamming vectors will generally offer better recall but will always require more vector elements, a larger memory, and more complicated postprocessing decoding. Regardless of the recollection vector choice, we have demonstrated that a pseudoinverse HAM, with no feedback as in iterative neural systems, offers impressive performance in the presence of noise.

Our results show the significant differences between HAMs and AAMs, the need to treat HAMs quite differently, and the importance of the use of proper performance measures. As associative memories become used in image analysis, image understanding and pattern recognition, HAMs must be employed. The new coding techniques advanced here show that such memories are attractive for these applications.

We acknowledge the support of this work by a grant from the Air Force Office of Scientific Research. We acknowledge the assistance of B.V.K. Vijaya Kumar of Carnegie Mellon University for assistance in the mathematical proofs included in the Appendices.

Appendix A: Proof of Theorem 1

The output vector is

$$\mathbf{y} = \mathbf{y}_k + \mathbf{M}\mathbf{n}. \tag{A1}$$

Substituting Eq. (A1) into the definition of σ_o^2 yields

$$\sigma_o^2 = E\{[(\mathbf{M}\mathbf{n})^i]^2\} = \sum_j \sum_k E\{m_{ij} m_{ik}\} E\{n^j n^k\}. \tag{A2}$$

Using the property of uncorrelated noise that $E\{n^j n^k\} = E\{[n^j]^2\}\delta_{jk}$, the definition $E\{[n^j]^2\} = \sigma_i^2$, and the independence of σ_i^2 from the summation indices (j and k), we obtain

$$\sigma_o^2 = \sigma_i^2 N E\{m_{ij}^2\}. \tag{A3}$$

Dividing both sides by σ_i^2, we obtain Theorem 1. This result is valid for any matrix whose key vectors are of dimension N and not just for the pseudoinverse matrix solution. Writing the squared Euclidean norm of M, we see[10] that the minimum norm solution is $\mathbf{M} = \mathbf{Y}\mathbf{X}^+$. It can also be shown that this solution is optimal for uncorrelated noise, and that it minimizes $E\{m_{ij}^2\}$ and also σ_o^2/σ_i^2 (i.e., the SNR ratio for the case of uncorrelated noise).

Appendix B: Proof of Theorem 2

For independent key vectors, the solution in Eq. (2) with $\mathbf{X}^+$ defined by Eq. (3) is valid, and thus for an AAM

$$\mathbf{M} = \mathbf{X}\mathbf{X}^+ = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T. \tag{B1}$$

The trace of $\mathbf{M}\mathbf{M}^T$ is

$$\mathrm{Tr}(\mathbf{M}\mathbf{M}^T) = \sum_i (\mathbf{M}\mathbf{M}^T)_{ii} = \sum_i \sum_j m_{ij}^2 = \mathrm{Tr}(\mathbf{M}), \tag{B2}$$

where the last equality follows from the fact that M is idempotent ($\mathbf{M} = \mathbf{M}^2$) and symmetric ($\mathbf{M} = \mathbf{M}^T$). The eigenvalues of an idempotent matrix are 0 or 1. The number of eigenvalues that are 1 is $r(\mathbf{M})$, i.e., the rank of M, and the trace satisfies $\mathrm{Tr}(\mathbf{M}) = r(\mathbf{M})$. To determine $r(\mathbf{M})$ for $\mathbf{M} = \mathbf{X}\mathbf{X}^+$, we first show that $r(\mathbf{X}) = M$ and that $r(\mathbf{X}^+) = M$. It then follows that $r(\mathbf{M}) = M = \mathrm{Tr}(\mathbf{M})$. Thus

$$\mathrm{Tr}[\mathbf{M}] = \sum_i \sum_j m_{ij}^2 = M. \tag{B3}$$

Using Eq. (B3), we prove Theorem 2:

$$E\{m_{ij}^2\} = \frac{\mathrm{Tr}(\mathbf{M})}{N^2} = \frac{M}{N^2}. \tag{B4}$$

Appendix C: Proof of Theorem 3

This follows directly by substituting Eq. (B4) into (A3).

Appendix D: Proof of Theorem 4

We consider Theorem 1, which applies for any matrix and derive an expression for $E\{m_{ij}^2\}$ for the HAM matrix written as $\mathbf{M} = \mathbf{Y}\mathbf{V}^{-1}\mathbf{X}^T$. We first rewrite Eq. (B2) for the general HAM case of recollection vectors of dimension K as

$$\mathrm{Tr}[\mathbf{M}\mathbf{M}^T] = \sum_i (MM^T)_{ii} = \sum_i \sum_j m_{ij}^2, \tag{D1}$$

where the summation over i runs from 1 to K and the summation over j runs from 1 to N. To evaluate the Theorem 1 equation for a HAM, we must obtain an expression for $E\{m_{ij}^2\}$. Letting the key vectors $\mathbf{x}_k$ (of dimension N) and the recollection vectors $\mathbf{y}_k$ (of dimension K) be random variables, we form the expected value of both sides of Eq. (D1) to obtain

$$E\{\mathrm{Tr}[\mathbf{M}\mathbf{M}^T]\} = \sum_i \sum_j E\{m_{ij}^2\}. \tag{D2}$$

The double summation in Eq. (D2) can be rewritten as

$$E\{\mathrm{Tr}[\mathbf{M}\mathbf{M}^T]\} = KNE\{m_{ij}^2\}. \tag{D3}$$

To evaluate Theorem 1 for this case and hence $E\{m_{ij}^2\}$, we require the trace of $\mathbf{M}\mathbf{M}^T$.

To obtain this, we substitute Eqs. (2) and (3) for an HAM into $\mathbf{M}\mathbf{M}^T$ and find

$$\mathbf{M}\mathbf{M}^T = \mathbf{Y}\mathbf{V}^{-1}\mathbf{Y}^T. \tag{D4}$$

The diagonal elements of the matrix product in Eq. (D4) are

$$(\mathbf{MM}^T)_{ii} = \sum_m \sum_k v_{mk}^{-1} y_{im} y_{ik}, \tag{D5}$$

where both summations are over the M vector pairs. The trace is the sum of (D5) over the diagonal elements ($i = 1$ to K) yielding

$$\mathrm{Tr}(\mathbf{MM}^T) = \sum_i \sum_m \sum_k v_{mk}^{-1} y_{im} y_{ik}. \tag{D6}$$

To evaluate Eq. (D3) and hence σ_o^2/σ_i^2, we form the expected value of both sides of Eq. (D6) and move the expected value operator within the summation as in Eq. (D2). With statistically uncorrelated key and recollection vectors, v_{mk}^{-1} and $y_{im}y_{ik}$ have no cross-correlations, and the expected value of their product is the product of their expected values. In practice, this assumption is not realistic, since the $\mathbf{y}_k$ depend on the $\mathbf{x}_k$ and are thus correlated (except for the case $\mathbf{Y} = \mathbf{I}$). In other tests,[3] each element of each $\mathbf{y}_k$ was chosen at random for the data that they used. Thus $E\{y_{im}y_{ik}\} = E\{y_{im}^2\}\delta_{km}$. This result is not valid for binary encoded $\mathbf{y}_k$ vectors but is valid for unit recollection vectors. With these assumptions,

$$E\{\mathrm{Tr}[\mathbf{MM}^T]\} = \sum_i \sum_m \sum_k E\{v_{mk}^{-1}\} E\{y_{im}^2\}\delta_{km}$$

$$= \sum_i \sum_m E\{v_{mm}^{-1}\} E\{y_{im}^2\}$$

$$= \sum_m E\{v_{mm}^{-1}\} \sum_i E\{y_{im}^2\}, \tag{D7}$$

where the last equality follows since $E\{v_{mm}^{-1}\}$ is independent of i. The second summation in Eq. (D7) is K times the expected value and $E\{y_{im}^2\}$ is independent of m (for the case of recollection vectors with equal power). This yields

$$E\{\mathrm{Tr}[\mathbf{MM}^T]\} = KE\{y_{im}^2\} E\{\mathrm{Tr}[\mathbf{X}^T\mathbf{X}]^{-1}\}. \tag{D8}$$

Substituting Eq. (D8) into Eq. (D3) and the result into Theorem 1, we prove Theorem 4.

References

1. T. Kohonen, *Self-Organization and Associative Memory* (Springer-Verlag, Berlin, 1988).
2. J. Hong and D. Psaltis, "Storage Capacity of Holographic Associative Memories," Opt. Lett. **11**, 812 (1986).
3. G. S. Stiles and D. L. Denq, "On the Effect of Noise on the Moore-Penrose Generalized Inverse Associative Memory," IEEE Trans. Pattern Anal. Machine Intell. **PAMI-7**, 358 (1985).
4. J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proc. Natl. Acad. Sci. U.S.A. **79**, 2554 (1982).
5. Y. Abu-Mostafa and J. St. Jacques, "Information Capacity of the Hopfield Model," IEEE Trans. Inf. Theory **IT-31**, 461 (1985).
6. G. Palm, "On Associative Memory," Biol. Cybern. **36**, 19 (1 Feb. 1980).
7. Y. Owechko, G. J. Dunning, E. Marom, and B. H. Soffer, "Holographic Associative Memory with Nonlinearities in the Correlation Domain," Appl. Opt. **26**, 1900 (1987).
8. K. Murakami and T. Aibara, "An Improvement on the Moore-Penrose Generalized Inverse Associative Memory," IEEE Trans. Syst. Man Cybern. **SMC-17**, 699 (1987).
9. C. L. Giles and T. Maxwell, "Learning, Invariance, and Generalization in High-Order Neural Networks," Appl. Opt. **26**, 4972 (1987).
10. C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and Its Applications* (Wiley, New York, 1971).
11. D. Casasent and B. Telfer, "Distortion-Invariant Associative Memories and Processors," Proc. Soc. Photo-Opt. Instrum. Eng. **697**, 60 (1986).
12. D. Casasent and B. Telfer, "Associative Memory Synthesis, Performance, Storage Capacity and Updating: New Heteroassociative Memory Results," Proc. Soc. Photo-Opt. Instrum. Eng. **848**, 313 (1987).
13. A. D. Fisher, W. L. Lippincott, and J. N. Lee, "Optical Implementations of Associative Networks with Versatile Adaptive Learning Capabilities," Appl. Opt. **26**, 5039 (1987).
14. B. Montgomery and B.V.K. Vijaya Kumar, "An Evaluation of the Use of the Hopfield Neural Network Model as a Nearest-Neighbor Algorithm," Appl. Opt. **25**, 3759 (1986).
15. D. Casasent, "Scene Analysis Research: Optical Pattern Recognition and Artificial Intelligence," Proc. Soc. Photo-Opt. Instrum. Eng. **634**, 439 (1986).
16. M. J. Little and C. S. Bak, "Enhanced Memory Capacity of a Hopfield Neural Network," Proc. Soc. Photo-Opt. Instrum. Eng. **698**, 150 (1986).
17. D. Psaltis and C. H. Park, "Nonlinear Discriminant Functions and Associative Memories," in *Neural Networks for Computing*, J. S. Denker, Ed. (American Institute of Physics, New York, 1986), p. 370.
18. D. Psaltis and N. Farhat, "Optical Information Processing Based on an Associative-Memory Model of Neural Nets with Thresholding and Feedback," Opt. Lett. **10**, 98 (1985).
19. J. A. Anderson, "Cognitive and Psychological Computation with Neural Models," IEEE Trans. Syst. Man Cybern. **SMC-13**, 799 (1983).
20. D. Casasent, "Unified Synthetic Discriminant Function Computational Formulation," Appl. Opt. **23**, 1620 (1984).
21. J. Goodman *et al.*, "Parallel Incoherent Optical Vector-Matrix Multiplier," Tech. Rep. L-723-1, BMD (Feb. 1979).
22. D. Casasent, J. Jackson, and C. Neuman, "Frequency-Multiplexed and Pipelined Iterative Optical Systolic Processors," Appl. Opt. **22**, 115 (1983).
23. D. Casasent and J. Jackson, "Space and Frequency-Multiplexed Optical Linear Algebra Processor," Appl. Opt. **25**, 2258 (1986).
24. K. Wagner and D. Psaltis, "A Space Integrating Acousto-Optic Matrix-Matrix Multiplier," Opt. Commun. **52**, 173 (1984).
25. G. H. Golub and C. Reinsch, "Singular Value Decomposition and Least Squares Solutions," Numer. Math. **14**, 403 (1970).
26. S. Liebowitz and D. Casasent, "Error-Correction Coding in an Associative Processor," Appl. Opt. **26**, 999 (1987).

CHAPTER 8:

"UPDATING OPTICAL PSEUDOINVERSE ASSOCIATIVE MEMORIES"

Submitted July 1988

UPDATING OPTICAL PSEUDOINVERSE ASSOCIATIVE MEMORIES

BRIAN TELFER AND DAVID CASASENT

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

Abstract

Selected algorithms for adding to and deleting from optical pseudoinverse associative memories are presented and compared. New realizations of pseudoinverse updating methods using vector inner product matrix bordering and reduced-dimensionality Karhunen-Loeve approximations (which have been used for updating optical filters) are described in the context of associative memories. Greville's theorem is reviewed and compared with the Widrow-Hoff algorithm. Kohonen's gradient projection method is expressed in a different form suitable for optical implementation. The data matrix memory is also discussed for comparison purposes. Memory size, speed and ease of updating, and key vector requirements are the comparison criteria used.

I. Introduction

A simple updating method is a desirable attribute of any associative memory (AM) model or architecture. By updating, we refer to both adding and deleting key/recollection pairs. Researchers primarily inspired by the operations of the brain do not usually consider deleting, since the brain does not explicitly delete items from its memory. However, deleting unused key/recollection pairs allows new pairs to be added without overloading conventional and realistic memories. Recently the neurally inspired Hopfield memory, based on a sum of vector outer products (VOP) matrix, has received much attention[1, 2, 3]. This model can be updated relatively easily, but suffers from poor recall accuracy (along with other problems)[4]. On the other hand, the pseudoinverse model of associative memory[5] can store key/recollection pairs

perfectly (meaning that if a key vector is input, the exact recollection vector is output) when the key vectors are linearly independent, and also is optimal for inputs that are key vectors distorted by additive white noise[6].

This paper will concentrate on pseudoinverse associative memories rather than other types of associative processors such as the Hopfield memory and its variations[1, 2, 3], higher-order correlation memories[7, 8], attentive memories[9], bidirectional memories[10, 11], holographic resonator memories[12, 13], etc. Our concern is with fast and efficient methods for updating a pseudoinverse memory (fast compared to directly recomputing the pseudoinverse). In comparing pseudoinverse updating algorithms intended for real-time or near real-time problems, speed must be the critical criterion. For these applications, we propose that the updating take place in dedicated hardware, either electronic or optical, and that the updated memory matrix then be loaded into an analog optical processor. An optical system is desirable for its high-speed parallel computing power. Simulations have shown that the pseudoinverse AM operates well even in systems with very low dynamic range[14], and thus an analog processor is sufficient, and attractive for its speed and simplicity. The amount of storage an updating algorithm requires and the conditions required on the key vectors in various pseudoinverse algorithms are also important issues that will be discussed. The accuracy of the final AM matrix is not of major concern since we envision using it in an analog optical processor. However, the accuracy of the operations performed to calculate the memory merit attention, since the final memory matrix must be useful. We consider updating the AM off-line on a digital processor. Thus, our concern is the time required to calculate the memory matrix (not the accuracy of the required calculations).

In Sec. II, we introduce our notation and briefly review the pseudoinverse memory. The two types of pseudoinverse memory optical architectures considered for recollection are a vector-inner-product (VIP) and a matrix-vector processor. These architectures are described in Sec. III. The matrices in these systems (especially in the matrix-vector architecture) can be computed by

various techniques. Our present concern is with methods to update these matrices. Sec. IV discusses the data matrix memory for comparison since this has the simplest of all updating methods. Sec. V presents an efficient and new VIP matrix bordering algorithm. A new reduced-dimensionality Karhunen-Loeve (KL) approximation to the pseudoinverse is discussed in Sec. VI. Sec. VII reviews Greville's theorem and briefly compares it to the Widrow-Hoff algorithm. A method that is related to Greville's theorem and which is based on gradient projection is considered in Sec. VIII. Sec. IX offers a summary and conclusion.

II. Pseudoinverse Memory Formation

Denoting the keys and recollections as vectors $\mathbf{x}_k$ (N-dimensional) and $\mathbf{y}_k$ (K-dimensional) respectively, where k = 1,...,M, the vectors $\mathbf{x}_k$ and $\mathbf{y}_k$ form an associated key/recollection pair (there are M such pairs). We desire a K×N matrix M satisfying

$$\mathbf{y}_k = \mathbf{M}\mathbf{x}_k, \quad \text{for } k = 1,...,M. \tag{1}$$

This notation also includes the special case of autoassociative memory (AAM), for which $\mathbf{x}_k$ and $\mathbf{y}_k$ are identical. Defining matrices $\mathbf{X}$ (N×M) and $\mathbf{Y}$ (K×M) with the key and recollection vectors as their columns, Eq. (1) can be rewritten as

$$\mathbf{Y} = \mathbf{M}\mathbf{X}. \tag{2}$$

The solution of Eq. (2) is

$$\mathbf{M} = \mathbf{Y}\mathbf{X}^{+}, \tag{3}$$

where $\mathbf{X}^{+}$ is the pseudoinverse[6] of $\mathbf{X}$. This solution minimizes the squared error $\|\mathbf{Y}-\mathbf{M}\mathbf{X}\|^2$. When the key vectors are linearly independent, the squared error is zero and Eq. (1) is solved exactly. Thus, when a key vector is input, the exact recollection is output. For the case of linearly independent keys,

$$\mathbf{X}^{+} = (\mathbf{X}^{T}\mathbf{X})^{-1}\mathbf{X}^{T}, \tag{4}$$

where the matrix $\mathbf{X}^T\mathbf{X}$ is the VIP matrix, since its ij-th element is the inner product $\mathbf{x}_i^T\mathbf{x}_j$. When the key vectors are orthonormal, $\mathbf{X}^T\mathbf{X} = \mathbf{I}$, and $\mathbf{X}^+ = \mathbf{X}^T$. In this case, the pseudoinverse AM simplifies to the VOP matrix $\mathbf{M} = \mathbf{Y}\mathbf{X}^T$.

In this paper, we will mainly be concerned with the case when the keys are linearly independent (a looser restriction than orthogonality). Thus, we require $M \leq N$, although it is possible for the keys to be linearly dependent even if $M < N$. In this case the minimum squared error (MSE) nature of the pseudoinverse solution is advantageous because the pseudoinverse yields (automatically) an approximate solution if an exact one does not exist. We are also primarily interested in applying heteroassociative memories to pattern recognition problems, in which case the recollection vectors are class labels and we can assume $K \ll M$.

III. Basic Optical Pseudoinverse Associative Memory Architectures

The simplest optical AM architecture is the standard optical matrix-vector multiplier shown in Fig. 1. The P_1 input plane contains N point modulators with light outputs proportional to the input $\mathbf{x}$. Each element of $\mathbf{x}$ uniformly illuminates one column of the matrix $\mathbf{M}$, a transmittance array of $K \times N$ elements at P_2, and the light leaving P_2 is integrated horizontally onto K separate detectors at P_3. The detector output is thus the matrix-vector product $\mathbf{y} = \mathbf{M}\mathbf{x}$. The memory matrix at P_2 is given by Eq. (3), where $\mathbf{X}^+$ can be obtained from Eq. (4) or by other algorithms to be discussed (Secs. V-VIII).

Although the key vectors are not orthonormal in general, the VIP optical architecture[15] of Fig. 2 represents an attractive implementation of the recall process when $\mathbf{X}^+ = \mathbf{X}^T$. The architecture consists of two cascaded matrix-vector multipliers, the first from P_1-P_3 and the second from P_3-P_5. The input $\mathbf{x}$ is loaded into point modulators at P_1 which illuminate the matrix $\mathbf{X}$ stored at P_2. The light leaving a slit placed at P_3 is the matrix-vector product $\mathbf{X}^T\mathbf{x}$. The P_3 output illuminates the matrix $\mathbf{Y}$ stored at P_4 and the output is the matrix-matrix-vector

multiplication $\mathbf{y} = \mathbf{Y}\mathbf{X}^T\mathbf{x}$, as desired. This architecture is attractive since the key/recollection vector pairs can be easily added or deleted by adding or deleting columns of $\mathbf{X}$ and $\mathbf{Y}$ at P_2 and P_4. In Sec. IV, we present a data matrix description of Fig. 2 and note that, with a maximum element selector at P_3, this system has no restrictions on the key vectors. In Sec. VIII, we discuss techniques for preprocessing key vectors so that the new transformed key vectors applied to the associative processor are orthonormal. With such preprocessing, the VIP architecture of Fig. 2 is attractive.

The VIP architecture requires storage of NM + KM matrix elements, whereas the standard matrix-vector processor requires one matrix with KN elements. The matrix-vector processor (with a pseudoinverse memory $\mathbf{M}$) requires fewer active matrix elements when $K < (NM)/(N-M)$. This condition is generally satisfied. For distortion-invariant AMs, it is often the case that a class of data is represented by several distorted (or other) versions, and that these key vectors all have the same recollection vectors. In this case, $K \ll M$. For iconic keys, we assume that $M < N$. Thus, we will assume throughout that $K \ll M < N$. In this case, the number of active memory elements required by the matrix-vector processor (Fig. 1) is much less than in the VIP architecture (Fig. 2). Thus, the ease of updating the VIP architecture must be weighed against its significantly larger spatial light modulator (SLM) space bandwidth product (SBWP) requirements. This provides motivation for attention to fast and efficient updating techniques for the pseudoinverse matrix-vector processor of Fig. 1.

For the case of an autoassociative memory ($\mathbf{Y} = \mathbf{X}$) with orthonormal key vectors ($\mathbf{X}^+ = \mathbf{X}^T$), the optical architecture of Fig. 2 can be reduced to a single matrix-vector system (Fig. 3) with optical flow from both left to right and right to left. In this bidirectional VIP architecture, P_2 stores $\mathbf{X}$, and P_1 and P_3 contain both detectors and point modulators. The input $\mathbf{x}$ is loaded into the N point modulators at P_1, and the matrix-vector product $\mathbf{X}^T\mathbf{x}$ is detected at P_3. This product is also the output from the point modulators at P_3 and is used to reilluminate P_2 (from

right to left). The final P_1 output $\mathbf{x}' = \mathbf{X}\mathbf{X}^T\mathbf{x}$ is read by the P_1 detectors. The same architecture has been proposed[11, 16, 17] for a bidirectional AM using a VOP matrix $\mathbf{M} = \mathbf{Y}\mathbf{X}^T$ with iterations in recall mode. Our realization is new since it implements a true pseudoinverse AM. Sec. VIII describes a technique to process nonorthogonal keys so that they can be stored in this architecture.

The final pseudoinverse memory architecture to be discussed follows from the formulation of M for the case of orthonormal key vectors. In this case $\mathbf{M} = \mathbf{Y}\mathbf{X}^T$ is the VOP AM and can be synthesized on-line optically[18, 19] on the system of Fig. 4. The architecture of Fig. 4 uses two crossed input transducers at P_1 and P_2 imaged onto a time integrating 2-D detector array at P_3, which can sum several VOP matrices. This is achieved by consecutively loading the $\mathbf{y}_k$ into P_1 and $\mathbf{x}_k$ into P_2 and summing their VOPs which form at P_3. This architecture is attractive since it achieves the memory synthesis optically. If P_3 in Fig. 4 is an optically addressed time-integrating SLM, then the VOP architecture of Fig. 4 can be combined with the matrix-vector system of Fig. 1 (where the P_3 SLM of Fig. 4 and the P_2 SLM of Fig. 1 are the same). With this combined system[19], memory synthesis and recall can both be achieved on-line optically. Sec. VIII considers preprocessing techniques that allow the architecture of Fig. 4 to be updated and to operate with nonorthogonal keys.

IV. Data Matrix

Before considering methods of updating the pseudoinverse memory, we consider the data matrix[4] as an associative memory because it can be updated very simply. The simplest data matrix memory is $\mathbf{M} = \mathbf{X}^T$. In recall mode, the output y is related to the input x by

$$\mathbf{y} = \mathbf{X}^T\mathbf{x}, \tag{5}$$

where the i-th element of y is the inner product of x and the key $\mathbf{x}_i$. Finding the maximum element of y identifies the key that is the nearest neighbor (in the Euclidean metric) of the

input, assuming the keys are normalized. General heteroassociative recall can be expressed as a data matrix memory such that

$$\mathbf{y} = \mathbf{Y}\Phi(\mathbf{X}^{\mathrm{T}}\mathbf{x}), \tag{6}$$

where $\Phi(\cdot)$ is a nonlinear operation that sets the maximum element of $(\cdot)$ to 1 and the other elements to 0. The single nonzero element causes the corresponding recollection to be output. No restrictions on the key vectors are necessary. Autoassociative operation can be achieved by setting $\mathbf{Y} = \mathbf{X}$ in Eq. (6).

The operation in Eq. (6) can be implemented by the VIP architecture of Fig. 2 with the nonlinearity $\Phi(\mathbf{X}^{\mathrm{T}}\mathbf{x})$ occurring at P_3. This nonlinearity can be realized by M detectors, followed by a maximum element selector, followed by M point modulators, one of which is selected to illuminate P_4. The maximum element selection can be implemented with a cascade of electronic comparators, or by a single stage of comparators with a varying threshold[19]. Winner-take-all networks[20, 21] can also be used to identify maximum elements, and it has been proposed[21] that a winner-take-all network be used at P_3 in the VIP architecture. With a maximum element selector at P_3 of Fig. 2, there are no restrictions on the key vectors. For an AAM, the single bidirectional matrix-vector system of Fig. 3 is sufficient, with the addition of the maximum selection nonlinearity $\Phi(\mathbf{X}^{\mathrm{T}}\mathbf{x})$ at P_3.

Thus, this data matrix description of the architectures of Figs. 2 and 3 is most attractive, since with a maximum element selector at P_3, these systems do not have any key vector restrictions. In addition, it is obvious that the data matrix memory can be easily updated. To add or delete key/recollection pairs, we simply add or delete columns of $\mathbf{X}$ and $\mathbf{Y}$ at P_2 and P_4 in Fig. 2, or for an AAM, add or delete columns of $\mathbf{X}$ at P_2 in Fig. 3. It is difficult to imagine a simpler updating scheme.

Although the pseudoinverse memory is more difficult to update than the data matrix, it does have several advantages. Among these are the facts that the pseudoinverse matrix-vector

architecture of Fig. 1 does not require a maximum element selector and that it requires only a small percentage of the active memory elements needed in the data matrix VIP architecture of Fig. 2, as shown in Sec. III. These remarks provide motivation for our attention to fast and efficient pseudoinverse memory update techniques.

V. Bordering Update Algorithm for Pseudoinverse Matrix

We first consider a method for updating the inverse VIP matrix in the pseudoinverse matrix expression in Eq. (4) using a bordering algorithm[22]. Once the inverse VIP matrix is updated, the memory matrix at P_2 of the simple matrix-vector processor of Fig. 1 can be quickly updated, as we will show. We rewrite Eq. (4) as $\mathbf{M}_M = \mathbf{Y}_M \mathbf{V}_M^{-1} \mathbf{X}_M^T$, where $\mathbf{V} = \mathbf{X}^T\mathbf{X}$ is the VIP matrix and where the subscript M indicates that the matrices are constructed from M key/recollection pairs. To update the memory by adding a new key/recollection pair ($\mathbf{x}_{M+1}$ and $\mathbf{y}_{M+1}$), we efficiently calculate $\mathbf{V}_{M+1}^{-1}$ from the prior $\mathbf{V}_M^{-1}$ without inverting a matrix. The algorithm is

$$\mathbf{V}_{M+1}^{-1} = \begin{bmatrix} \mathbf{V}_M^{-1} + \frac{1}{\alpha}\mathbf{v}\mathbf{v}^T & -\frac{1}{\alpha}\mathbf{v} \\ -\frac{1}{\alpha}\mathbf{v}^T & \frac{1}{\alpha} \end{bmatrix} \tag{7}$$

where $\mathbf{v} = \mathbf{V}_M^{-1}\mathbf{X}_M^T\mathbf{x}_{M+1}$, and $\alpha = \mathbf{x}_{M+1}^T\mathbf{x}_{M+1} - \mathbf{x}_{M+1}^T\mathbf{X}_M\mathbf{v}$. Then the new memory can be calculated from Eqs. (3) and (4), given the new inverse VIP matrix in Eq. (7).

This is the standard bordering algorithm. When its computational load is evaluated, we find that the faster time to calculate $\mathbf{V}^{-1}$ yields only a small improvement in the overall time to update $\mathbf{M}$ (whose computation is still dominated by the required matrix-matrix multiplications). Thus, we devised a new implementation that is much faster. We now detail this

implementation. We first partition $\mathbf{X}_{M+1}$ and $\mathbf{Y}_{M+1}$ as

$$\mathbf{X}_{M+1} = \left[\mathbf{X}_M \quad \mathbf{x}_{M+1}\right] \tag{8}$$

and

$$\mathbf{Y}_{M+1} = \left[\mathbf{Y}_M \quad \mathbf{y}_{M+1}\right] . \tag{9}$$

Using Eqs. (4), (7), and (8), we rewrite ${\mathbf{X}_{M+1}}^{+}$ as

$${\mathbf{X}_{M+1}}^{+} = \begin{bmatrix} {\mathbf{X}_M}^{+} + \frac{1}{\alpha}\mathbf{v}\mathbf{v}^T{\mathbf{X}_M}^T - \frac{1}{\alpha}\mathbf{v}{\mathbf{x}_{M+1}}^T \\ \\ -\frac{1}{\alpha}\mathbf{v}^T{\mathbf{X}_M}^T + \frac{1}{\alpha}{\mathbf{x}_{M+1}}^T \end{bmatrix}. \tag{10}$$

Then using Eqs. (3), (9) and (10), the new memory is expressed as

$$\begin{aligned} \mathbf{M}_{M+1} = \quad & \mathbf{M}_M + \frac{1}{\alpha}\mathbf{Y}_M\mathbf{v}\mathbf{v}^T{\mathbf{X}_M}^T - \frac{1}{\alpha}\mathbf{Y}_M\mathbf{v}{\mathbf{x}_{M+1}}^T - \\ & \frac{1}{\alpha}\mathbf{y}_{M+1}\mathbf{v}^T{\mathbf{X}_M}^T + \frac{1}{\alpha}\mathbf{y}_{M+1}{\mathbf{x}_{M+1}}^T. \end{aligned} \tag{11}$$

This new algorithm does not form the new ${\mathbf{V}_{M+1}}^{-1}$, rather it uses only $\mathbf{v}$ and α to compute the new $\mathbf{M}_{M+1}$. The updating steps are summarized in the flowchart of Fig. 5.

Prior bordering algorithms only considered adding a new vector pair. We now detail a new algorithm by which the bordering technique can be used to delete a key/recollection pair from the memory. Consider the original memory containing M+1 vector pairs. The key step is to compute ${\mathbf{V}_M}^{-1}$ from ${\mathbf{V}_{M+1}}^{-1}$. We first consider how to delete the (M+1)-th pair, and then extend the algorithm to deleting any vector pair. First, we rewrite ${\mathbf{V}_{M+1}}^{-1}$ as the four partitions

$${\mathbf{V}_{M+1}}^{-1} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \\ \mathbf{b}^T & c \end{bmatrix} \tag{12}$$

where $\mathbf{A}$ is M×M, $\mathbf{b}$ is M×1 and c is a scalar. By comparing Eqs. (7) and (12), it is apparent that $\mathbf{A} = \mathbf{V}_M^{-1} + (1/\alpha)\mathbf{v}\mathbf{v}^T$, $\mathbf{b} = -(1/\alpha)\mathbf{v}$, and $c = 1/\alpha$. From these expressions, we find our desired result to be

$$\mathbf{V}_M^{-1} = \mathbf{A} - \frac{1}{c}\mathbf{b}\mathbf{b}^T. \tag{13}$$

The new memory will then be $\mathbf{M}_M = \mathbf{Y}_M\mathbf{V}_M^{-1}\mathbf{X}_M^T$. To delete the k-th vector pair, where $k \neq M+1$, we first make it the (M+1)-th pair, so we can use the deletion bordering expressions in Eqs. (12) and (13). It is easy to show that this is achieved by moving the k-th row and column of $\mathbf{V}_{M+1}^{-1}$ to the (M+1)-th row and column. We then partition $\mathbf{V}_{M+1}^{-1}$ as in Eq. (12) and compute $\mathbf{V}_M^{-1}$ from Eq. (13). These steps are summarized in the flowchart of Fig. 6.

Note that both the adding and deleting algorithms require storage of the matrices $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{V}$. We also note that both bordering updating methods require the key vectors to be linearly independent. We now consider the number of operations required by these updating algorithms.

The number of floating point multiplications needed to add a vector pair to the pseudoinverse memory using our improved bordering algorithm is given in Table I. Table I includes only those operations that, depending on the value of K, can contribute significantly to the total computational load. The dominant terms under the assumption $K \ll M < N$ are denoted by an asterisk in Table I and in subsequent tables. From Table I, we find that our new algorithm requires only $2NM + M^2$ multiplications. To compute $\mathbf{M}_{M+1}$ directly from Eqs. (3) and (4), without using bordering, requires approximately $NM^2 + KNM + (1/2)M^3$ multiplications (with the last term[23] due to the direct inversion of $\mathbf{V}$). From Table I, it is clear that our version of the bordering algorithm is much faster. Most of this speed improvement is due to our new partitioning technique for the key and recollection matrices and not to the bordering method for computing the new $\mathbf{V}^{-1}$. To see this, we note that the number of multiplications required to update $\mathbf{V}^{-1}$ by the bordering algorithm in Eq. (7) is $O(M^2)$ compared

to $O(M^3)$ for direct inversion. If the new memory matrix were computed by first obtaining $\mathbf{V}^{-1}$ in Eq. (7) and then performing the multiplication $\mathbf{Y}\mathbf{V}^{-1}\mathbf{X}^T$,the second matrix-matrix multiplication is seen to require NM^2 multiplications. This equals over half the total number of multiplications required for direct computation. Thus, obtaining $\mathbf{V}^{-1}$ from Eq. (7) and then computing the new M will still result in $O(NM^2)$ multiplications. This is a factor of M more multiplications than we require in our technique. Thus, our algorithm for partitioning the key and recollection matrices is essential. Numerical values for the speed differences between the various algorithms are given in Sec. IX.

To delete a vector pair requires $NM^2 + KNM + 2M^2$ multiplications, which is much more than our new algorithm requires for adding a vector, but is less than that for directly recomputing the memory. The speed difference between adding and deleting vector pairs exists because we have not found a partitioning approach to compute $\mathbf{M}_M$ from the elements of $\mathbf{V}_M^{-1}$ for the case of deleting vector pairs.

If a highly parallel architecture such as the PRIMO analog optical processor[24] were fully developed, then it might be attractive to directly compute $\mathbf{M}_{M+1}$ by using Faddeev's algorithm[22] which computes $\mathbf{V}_{M+1}^{-1}$ and the two matrix-matrix multiplications in the expression $\mathbf{M}_{M+1} = \mathbf{Y}_{M+1}\mathbf{V}_{M+1}^{-1}\mathbf{X}_{M+1}^T$. The PRIMO processor completes these computations in 4(N+M) clock cycles[24]. The usefulness of the result obtained with such an analog optical processor remains to be addressed. Since a survey of all parallel processors is beyond the scope of this paper, we use the number of multiplications as a useful comparison measure for different algorithms.

VI. KL Pseudoinverse Approximation and Updating

In applications such as pattern recognition, we are concerned with distortion-invariant HAMs. In such cases, the key vectors can be grouped into classes (with the key vectors within each class being distorted versions of each other, or different types of the same class). In the usual case, all key vectors of the same class will have the same recollection vectors (but this is not a requirement for the following method). In such applications, we can adapt a near real-time pattern recognition algorithm[25] to calculate the memory matrix. In this method, computations are performed in a reduced dimensionality hyperspace (hence the algorithm is near real-time) produced by the Karhunen-Loeve (KL) transform[26]. Hence, we refer to this as a KL approximation and update method. The result is then transformed to the full dimensionality space to obtain the memory M.

The key step in this algorithm is the calculation of the eigenvectors of the correlation matrix from the eigenvectors of the VIP matrix. We first show how this is achieved and how it is faster than computing the eigenvectors of the correlation matrix when $M_1 < N$ (where the subscript 1 denotes that this is the correlation matrix of the key vectors for a single class of data). The correlation matrix $\mathbf{R}_1 = (1/M_1)\mathbf{X}_1\mathbf{X}_1^T$ is N×N, but its rank is only M_1 (or less, if the keys are dependent). A large effort is required to calculate the eigenvectors of $\mathbf{R}_1$. However, it has been shown[27] that the eigenvectors of $\mathbf{R}_1$ can be computed from the eigenvectors of the scaled VIP matrix $\mathbf{V}_1 = (1/M_1)\mathbf{X}_1^T\mathbf{X}_1$. This is much easier, since $\mathbf{V}_1$ is only of size $M_1 \times M_1$. Once we have calculated the M_1 eigenvectors ψ_k of $\mathbf{V}_1$, we can obtain the M_1 eigenvectors ϕ_k of $\mathbf{R}_1$ from the scalar-vector sum

$$\phi_k = \sum_{i=1}^{M_1} \psi_k^i \mathbf{x}_i, \tag{14}$$

where the subscripts k denote specific eigenvectors, the superscript denotes a particular element within the vector ψ_k, and the $\mathbf{x}_i$ are the key vectors. Each ϕ_k (of size N×1) is thus a linear

combination of the key vectors $\mathbf{x}_i$ with the weighting coefficients determined by the elements of each ψ_k. The M_1 eigenvectors of $\mathbf{R}_1$ that have nonzero eigenvalues (or fewer if there are linearly dependent keys) can be calculated in this way. The remaining N - M_1 eigenvectors of $\mathbf{R}_1$ (or more if there are linearly dependent keys) have zero eigenvalues and are not needed. The ϕ_k can be normalized by dividing them by $\sqrt{M_1 \lambda_k \psi_k^T \psi_k}$ (where λ_k are the eigenvalues of the ψ_k, which are the same as the eigenvalues of the ϕ_k).

To use this VIP eigenvector algorithm, we first compute the VIP matrix for each class and the eigenvectors of the **R** for each class. In order to reduce dimensionality, we retain only the η dominant eigenvectors from each class. This is the KL approximation. Only the few most dominant eigenvectors from each class are needed to accurately represent each class. The total number of retained eigenvectors is then ηC, where C is the number of classes, and $\eta C \ll M$ (M is the total number of key vectors for all classes). In the original algorithm[25], the ϕ_k were orthogonalized by the Gram-Schmidt technique. However, this is not necessary. We now discuss how to obtain **M** from the ϕ_k, where k = 1,...,ηC.

We form the N$\times\eta$C matrix $\tilde{\mathbf{X}}$ which has the eigenvectors ϕ_k as its columns. Then **X** is transformed into the reduced dimensionality space by

$$\mathbf{A} = \tilde{\mathbf{X}}^{\mathrm{T}}\mathbf{X}, \tag{15}$$

where **A** is of size ηC$\times$M. We first formulate the associative memory as a K$\times\eta$C matrix **E** in this reduced space satisfying

$$\mathbf{EA} = \mathbf{Y}. \tag{16}$$

This **E** is the reduced dimensionality space representation of the full **M** (of size K$\times$N). We then solve for **E** by

$$\mathbf{E} = \mathbf{YA}^{+}. \tag{17}$$

Since **A** is of size ηC$\times$M and $\eta C \ll M$, it is reasonable to assume that its rows are independent.

This allows us to obtain $\mathbf{A}^+$ by the formula[5]

$$\mathbf{A}^+ = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}. \tag{18}$$

It may be possible to update $\mathbf{A}^+$ in Eq. (18) rather than directly computing it. We have not considered such methods, because the main computational load for this method is the matrix-matrix multiplication in Eq. (15) and the matrix inversion in Eq. (18) is simple since $\mathbf{A}\mathbf{A}^T$ is a small matrix. Using Eq. (18), we obtain $\mathbf{E}$ in Eq. (17) and then obtain the final $\mathbf{M}$ (in terms of the key vectors in the original space) using

$$\mathbf{M} = \mathbf{E}\tilde{\mathbf{X}}^T. \tag{19}$$

The resulting $\mathbf{M}$ will not be the exact pseudoinverse memory, but will be a very close and useful approximation to it. The close approximation is due to the fact that the KL transform is the optimum (MSE) method of representing a single class in a reduced dimension[26]. We KL transform each class separately, because the KL transform is not expected to accurately represent more than one class with a small number of eigenvectors. Because the KL transform's eigenvector representation is used, this algorithm and solution can be applied in cases when the key vectors are linearly dependent. This makes it quite general and attractive.

Next, we summarize an iterative procedure[27] to compute the best η eigenvectors for each class. This procedure is even more efficient than the VIP matrix method discussed above, and can also be used for updating $\mathbf{M}$.

We first form the scaled VIP matrix (scaled, because it is divided by the number of vectors forming it) for the first $\eta+1$ key vectors of one class. We calculate its dominant η eigenvectors, using the method discussed above. This is simple, since η is typically[27] less than 10 and the VIP matrix is thus less than 11×11. The remaining key vectors of the same class are input sequentially, one per iteration. The iteration index k is initially $\eta+2$. In the following description, we use two subscripts. The first denotes the iteration number and the second

denotes the number of a vector within a set. At iteration k, the previous V of size $(\eta+1)\times(\eta+1)$ is updated using the new key vector $\mathbf{x}_k$ and the η eigenvectors $\boldsymbol{\phi}_{k-1,i}$ (i = 1,...,η) from the previous k-1 iteration. The update algorithm for the elements (i, j) of $\mathbf{V}_k$ at the k-th iteration is:

$$
\begin{aligned}
(\mathbf{V}_k)_{i,j} &= \frac{k-1}{k}(\lambda_{k-1,i}\lambda_{k-1,j})^{1/2}\delta_{ij} \qquad i,j = 1,\ldots,\eta \\
(\mathbf{V}_k)_{i,\eta+1} &= (\mathbf{V}_k)_{\eta+1,i} = \frac{1}{k}\boldsymbol{\phi}_{k-1,i}{}^T\mathbf{x}_k \qquad i = 1,\ldots,\eta \\
(\mathbf{V}_k)_{\eta+1,\eta+1} &= \frac{1}{k}\mathbf{x}_k{}^T\mathbf{x}_k.
\end{aligned} \tag{20}
$$

For i,j = 1,...,η, $(\mathbf{V}_k)_{i,j}$ is a scaled version of the VIP $\boldsymbol{\phi}_{k-1,i}{}^T\boldsymbol{\phi}_{k-1,j}$. Since these eigenvectors are orthogonal, the nondiagonal elements of $\mathbf{V}_k$ for i,j = 1,...,η are zero, as indicated by the first line of Eq. (20). The first η elements of the last row and column of $\mathbf{V}_k$ are scaled VIPs of the eigenvectors $\boldsymbol{\phi}_{k-1,i}$ and the new key $\mathbf{x}_k$. The element $(\mathbf{V}_k)_{\eta+1,\eta+1}$ is a scaled version of the VIP of the new key with itself. These elements are expressed in the second and third lines of Eq. (20). At each iteration, $\mathbf{V}_k$ remains $(\eta+1)\times(\eta+1)$ and incorporates the best η components of the previous k-1 key vectors with the new key. The η newest and best eigenvectors $\boldsymbol{\phi}_{k,i}$ (i = 1,...,η) (in terms of the original key vector space) and their eigenvalues $\lambda_{k,i}$ are found from the eigenvectors $\boldsymbol{\psi}_{k,i}$ and eigenvalues $\lambda_{k,i}$ of $\mathbf{V}_k$ and the new key vector $\mathbf{x}_k$ using the recursive formula

$$
\boldsymbol{\phi}_{k,i} = \psi_{k,i}{}^1\boldsymbol{\phi}_{k-1,1} + \psi_{k,i}{}^2\boldsymbol{\phi}_{k-1,2} + \cdots \psi_{k,i}{}^\eta\boldsymbol{\phi}_{k-1,\eta} + \psi_{k,i}{}^{\eta+1}\mathbf{x}_k \tag{21}
$$

where

$$
\boldsymbol{\phi}_{k,i}{}^T\boldsymbol{\phi}_{k,i} = k\lambda_{k,i}, \quad \boldsymbol{\psi}_{k,i}{}^T\boldsymbol{\psi}_{k,i} = 1. \tag{22}
$$

The new $\boldsymbol{\phi}_k$ are orthogonal and are a linear combination of all key vectors $\mathbf{x}_k$ through the present iteration.

Eq. (21) approximates $\boldsymbol{\phi}_{k,i}$ by a linear combination with only $\eta+1$ coefficients. This is less

than the k coefficients that are needed for an exact representation, but the approximation error is small, as stated earlier. Recall that η is a fixed constant and k is the total number of key vectors seen thus far. The algorithm iterates until $k = M_I$ (the number of key vectors in a single class) and the output is the best η eigenvectors. These eigenvectors are then normalized using Eq. (22). This procedure is performed for each class, and then the memory **M** is calculated using Eqs. (15) - (19), with the ϕ_k now being determined by the algorithm in Eqs. (20) - (22).

To use this iterative technique for updating the pseudoinverse memory matrix (i.e., if we want to add a new key vector $\mathbf{x}_{M+1}$ to an already existing class), we input the new $\mathbf{x}_{M+1}$, update the eigenvectors for that class using Eqs. (20) - (22) and then recalculate **M** using Eqs. (15) - (19). Since only the eigenvectors of one class have changed, we can use the projection values previously calculated in Eq. (15) for the projections of the $\mathbf{x}_k$ in $\mathbf{X}_M$ onto the unchanged (η-1)C eigenvectors ϕ_k in $\tilde{\mathbf{X}}$. Also, the matrix $\mathbf{AA}^T$ in Eq. (18) needs to be modified using only the new elements of **A** (a small percentage of the total matrix). In this new realization of the KL updating algorithm, the computation time is considerably reduced by using these observations. The algorithm requires storage of the ηC eigenvalues (where $\eta C \ll M$), and the matrices **A**, $\mathbf{AA}^T$, $\tilde{\mathbf{X}}$, **X** and **Y**. Because this method does not store the entire M×M VIP matrix as is required in the bordering method, it uses less storage than the bordering method. The KL algorithm also has the major practical advantage over the bordering algorithm that it can operate on dependent keys.

The number of multiplications needed to add a vector pair (assuming it is a member of a class already stored in the memory) using the iterative KL algorithm is given in Table II. The major computational load in Table II is the two terms ηNM and ηCKN. The term ηNM arises from transforming **X** into the reduced dimensional space in Eq. (18). If the existing projection values were not used, this operation would require ηCNM multiplications. Thus, using the existing projection values, as we suggest, results in a considerable reduction in computation time.

The two dominant terms in Table II will normally be larger (by a factor of about 3) than the bordering algorithm's two dominant terms 2NM and M^2 (from Table I). However, the KL algorithm and all other algorithms that we address are roughly of O(NM) in complexity, when their implementations are optimized by the methods we advance. Quantitative numerical comparisons are provided in Sec. IX.

To delete a vector pair, we need to recompute the η eigenvectors ϕ_k for that class and then recompute M using Eqs. (15) - (19). This procedure requires the same number of multiplications as for adding a vector pair, plus $(1/C)\eta^2$NM (to recompute the η eigenvectors). This additional term is nonnegligible, but the computation time is of the same order of magnitude as required to add a vector pair using the KL algorithm.

VII. Pseudoinverse Matrix Updating Using Greville and Widrow-Hoff Algorithms

This section reviews Greville's theorem[5] and briefly compares it with the matrix form of the Widrow-Hoff algorithm[18]. Sec. VIII then discusses the related gradient projection method[5]. Greville's theorem for updating a pseudoinverse matrix is

$$\mathbf{X}_{M+1}{}^{+} = \begin{bmatrix} \mathbf{X}_M{}^{+}(\mathbf{I} - \mathbf{x}_{M+1}\boldsymbol{\rho}^T) \\ \boldsymbol{\rho}^T \end{bmatrix}, \tag{23}$$

where the vector

$$\boldsymbol{\rho} = \begin{cases} \dfrac{(\mathbf{I} - \mathbf{X}_M\mathbf{X}_M{}^{+})\mathbf{x}_{M+1}}{\|(\mathbf{I} - \mathbf{X}_M\mathbf{X}_M{}^{+})\mathbf{x}_{M+1}\|^2} & \text{if the numerator } \neq 0, \\ \dfrac{(\mathbf{X}_M{}^{+})^T\mathbf{X}_M{}^{+}\mathbf{x}_{M+1}}{1 + \|\mathbf{X}_M{}^{+}\mathbf{x}_{M+1}\|^2} & \text{otherwise.} \end{cases} \tag{24}$$

Greville's theorem can be used for linearly dependent keys, in which case the second definition of $\boldsymbol{\rho}$ applies. Once $\boldsymbol{\rho}$ has been calculated, the memory matrix is updated by[5]

$$\mathbf{M}_{M+1} = \mathbf{M}_M + (\mathbf{y}_{M+1} - \mathbf{M}_M\mathbf{x}_{M+1})\boldsymbol{\rho}^T. \tag{25}$$

Although Eq. (23) is not needed to compute the new memory matrix, it is needed to update $\mathbf{X}_M{}^+$ to $\mathbf{X}_{M+1}{}^+$, which is required for the next vector pair being added.

The iterative Widrow-Hoff algorithm is

$$\mathbf{M}_{n+1} = \mathbf{M}_n + \alpha(\mathbf{y}_{n+1} - \mathbf{M}_n\mathbf{x}_{n+1})\mathbf{x}_{n+1}{}^T, \tag{26}$$

where α is a scale factor and the subscript denotes the iteration. This is similar to the form of the Greville algorithm in Eq. (25) with the exception that $\boldsymbol{\rho}^T$ in Eq. (25) is a matrix times $\mathbf{x}_{M+1}{}^T$, where the matrix has been chosen to produce the solution $\mathbf{M}_{M+1} = \mathbf{Y}_{M+1}\mathbf{X}_{M+1}{}^+$ in a single step. Conversely, the Widrow-Hoff algorithm normally uses a predetermined scale factor α and iterates until a solution is reached. To compare the speed of these two algorithms, we observe that Greville's theorem requires roughly 3NM multiplications to compute $\boldsymbol{\rho}$ and $\mathbf{X}_{M+1}{}^+$ (for linearly independent keys), while Widrow-Hoff requires 2KN multiplications per iteration. Therefore Widrow-Hoff requires more multiplications than Greville's theorem if there are more than (3NM)/(2KN) = 3M/2K Widrow-Hoff iterations. Widrow-Hoff must iterate through each of the M+1 vector pairs at least once (i.e. a minimum of M iterations). This is more than 3M/2K (when $K \geq 2$, which is normally the case). Thus, we expect Widrow-Hoff to require at least 2K/3 times the number of multiplications needed in Greville's theorem, although the exact speed difference is not easily quantified. An analog optical updating architecture that uses the Widrow-Hoff algorithm is being developed[18]. We note that the above speed argument on whether the Greville algorithm is faster than the Widrow-Hoff algorithm (in terms of the multiplications required, not cycles of the system) does not apply directly to this optical architecture, which is parallel.

Table III lists the number of multiplications for Greville's theorem to add a vector pair to

the pseudoinverse memory. The result from Table III (dominant terms only) is either 3NM or 4NM operations (depending on whether the key vectors are linearly independent or dependent). Both terms are likely to be smaller than the term ηNM in the KL algorithm (since η>4). Thus, we expect Greville's theorem will be faster than the KL algorithm, although both will be of the same order of magnitude. Because the bordering algorithm's dominant terms are 2NM and M^2, it is faster than Greville's theorem. There does not exist an efficient method for deleting vector pairs with Greville's theorem. Thus it is slightly less attractive. Greville's method requires that the matrices $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{X}^+$ and $\mathbf{M}$ be stored. Since the matrix $\mathbf{X}^+$ contains NM elements, Greville's theorem requires about twice the storage of the bordering and KL methods. Numerical values for the speed differences between the various algorithms are given in Sec. IX.

The Widrow-Hoff algorithm requires storage of the matrices $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{M}$. This is the smallest amount of storage for the five updating algorithms that we consider (the savings is very small however). With Widrow-Hoff, it is trivial to delete a vector pair from $\mathbf{X}$ and $\mathbf{Y}$, but deleting a pair from the memory matrix only occurs gradually as more vector pairs are added to the memory (this has been referred to as the "forgetting" effect[18]).

VIII. Updating Using Gradient Projection AMs

Kohonen[5] developed the gradient projection method from Greville's theorem as a "computational scheme" for finding the pseudoinverse memory output without explicitly computing a memory matrix. We first detail the algorithm and then discuss new implementation considerations for forming a memory matrix so that the recall operation can be realized on an optical matrix-vector processor.

For the case of an AAM, the new algorithm we developed for VIP implementation first orthogonalizes the key vectors using the Gram-Schmidt (GS) algorithm[28]. This generates orthonormal vectors $\hat{\mathbf{x}}_k$ from the key vectors $\mathbf{x}_k$. Thus, this algorithm is suitable for any key vectors (not necessarily even linearly independent key vectors). The first orthonormal vector, $\hat{\mathbf{x}}_1$,

is formed as

$$\hat{\mathbf{x}}_1 = \mathbf{x}_1 / \|\mathbf{x}_1\|. \tag{27}$$

Thereafter, the k-th orthonormal vector $\hat{\mathbf{x}}_k$ is obtained from the prior k-1 orthonormal vectors $\hat{\mathbf{x}}_i$ and the new $\mathbf{x}_k$ key vector using

$$\hat{\mathbf{x}}_k = \{\mathbf{x}_k - \sum_{i=1}^{k-1} (\mathbf{x}_k{}^T \hat{\mathbf{x}}_i)\hat{\mathbf{x}}_i\} \; / \; \|\mathbf{x}_k - \sum_{i=1}^{k-1} (\mathbf{x}_k{}^T \hat{\mathbf{x}}_i)\hat{\mathbf{x}}_i\|. \tag{28}$$

Eqs. (27) and (28) are used only for those $\hat{\mathbf{x}}_k$ with nonzero norms. If any $\hat{\mathbf{x}}_k$ equals **0** (which will occur if $\mathbf{x}_k$ is linearly dependent), it is discarded. Eqs. (27) and (28) differ from the expressions given by Kohonen[5] in that we normalize the vectors $\hat{\mathbf{x}}_k$ as they are produced. This reduces the number of computations required. To use these $\hat{\mathbf{x}}_i$, the AAM output $\hat{\mathbf{x}}$ due to input $\mathbf{x}$ is

$$\hat{\mathbf{x}} = \sum_{i=1}^{M} (\mathbf{x}^T \hat{\mathbf{x}}_i)\hat{\mathbf{x}}_i. \tag{29}$$

We rewrite Eq. (29) as

$$\hat{\mathbf{x}} = \hat{\mathbf{X}}\hat{\mathbf{X}}^T\mathbf{x}, \tag{30}$$

where $\hat{\mathbf{X}}$ has the $\hat{\mathbf{x}}_k$, k = 1,...,M, as its columns. From this new description in Eq. (30) in terms of orthonormalized key vectors, it is clear that the AAM matrix used is $\mathbf{M} = \hat{\mathbf{X}}\hat{\mathbf{X}}^T$. We derived this new form in Eq. (30) to show that this computed $\hat{\mathbf{X}}$ can be stored at P_2 of Fig. 3. Because the key vectors have been orthogonalized by preprocessing, the original key vectors are no longer required to be orthonormal. Thus, no maximum element selector is necessary, significantly simplifying this architecture. From our new form in Eq. (30), the matrix M can also be stored in a single matrix-vector multiplication architecture (Fig. 1) as the sum of VOPs $\mathbf{M} = \sum_{k=1}^{M} \hat{\mathbf{x}}_k \hat{\mathbf{x}}_k{}^T$ computed on the system of Fig. 4 (without the prior requirement of orthonormal key vectors). Thus, this algorithm realistically and significantly simplifies all possible optical architectures. Note that since the updating methods in Secs. V-VII have also been formulated as VOP operations, they can also be implemented on the system of Fig. 4. For an AAM, the

VOP architecture requires a larger matrix than the bidirectional VIP architecture of Fig. 3 (N×N vs. N×M, where M<N for a pseudoinverse AAM). Since the same gradient projection updating method can be used on the architectures of Figs. 1 and 3, the VIP architecture is preferable for a pseudoinverse AAM.

We now consider an HAM. For this, Kohonen[5] showed that the $\hat{\mathbf{x}}_k$ are calculated as before in Eqs. (27) and (28) with the additional calculation of modified recollection vectors $\hat{\mathbf{y}}_k$ given by

$$\hat{\mathbf{y}}_1 = \mathbf{y}_1 \tag{31}$$

$$\hat{\mathbf{y}}_k = \mathbf{y}_k - \sum_{i=1}^{k-1} (\mathbf{x}_k{}^T \hat{\mathbf{x}}_i)\hat{\mathbf{y}}_i. \tag{32}$$

The output $\mathbf{y}$ for an input $\mathbf{x}$ is then

$$\mathbf{y} = \sum_{i=1}^{M} (\mathbf{x}^T \hat{\mathbf{x}}_i)\hat{\mathbf{y}}_i. \tag{33}$$

We now rewrite this equation as a new matrix-matrix-vector multiplication

$$\mathbf{y} = \hat{\mathbf{Y}}\hat{\mathbf{X}}^T\mathbf{x} \tag{34}$$

where $\hat{\mathbf{Y}}$ and $\hat{\mathbf{X}}$ have the $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{y}}_i$ respectively as their columns. This new form in Eq. (34) is suitable for realization on the matrix-vector VIP architecture of Fig. 2 or on the VOP architecture of Fig. 4. This rewritten algorithm is attractive because it places no restriction on the key vectors. However, if a key vector is linearly dependent, its recollection vector must be the corresponding linear combination of the other recollection vectors. This requirement is met automatically by AAMs.

We now detail how to update this memory (i.e. how to add or delete a key/recollection vector pair, or substitute one vector pair for another). The following discussion considers how to update an HAM (updating an AAM is simpler, because only the $\hat{\mathbf{x}}_k$ and not the $\hat{\mathbf{y}}_k$ need to be recomputed). We first discuss how to add a vector pair. The new vectors $\hat{\mathbf{x}}_{M+1}$ and $\hat{\mathbf{y}}_{M+1}$ are computed from $\mathbf{x}_{M+1}$ and $\mathbf{y}_{M+1}$ by Eqs. (28) and (32) with the substitution k=M+1. Then the outer product $\hat{\mathbf{y}}_{M+1}\hat{\mathbf{x}}_{M+1}{}^T$ is added to the VOP memory optically or the new $\hat{\mathbf{x}}_{M+1}$ and $\hat{\mathbf{y}}_{M+1}$

column vectors are added to P_2 and P_4 in the VIP architecture of Fig. 2. To delete a vector pair, x_k and y_k, from the memory, all vectors $\hat{x}_{k+1},...,\hat{x}_M$ and $\hat{y}_{k+1},...,\hat{y}_M$ following the vector to be deleted must be recomputed by Eqs. (28) and (32) (since they are no longer required to be orthonormal to $\hat{x}_k$ and $\hat{y}_k$). The old vectors $\hat{x}_{k+1},...,\hat{x}_M$ and $\hat{y}_{k+1},...,\hat{y}_M$ in M are replaced by the recomputed vectors. The first k-1 vectors $\hat{x}_1,...,\hat{x}_{k-1}$ and $\hat{y}_1,...,\hat{y}_{k-1}$ do not need to be altered since $\hat{x}_k$ and $\hat{y}_k$ were not used in calculating them. The computational complexity to delete a vector thus ranges from no calculations to delete the last vector pair (x_M,y_M) to rerunning the entire Gram-Schmidt algorithm to delete the first vector pair (x_1,y_1). To substitute the k-th vector pair with another pair: the old pair is first discarded, $\hat{x}_k$ and $\hat{y}_k$ are computed for the new pair, then $\hat{x}_{k+1},...,\hat{x}_M$ and $\hat{y}_{k+1},...,\hat{y}_M$ are recomputed so they will be orthonormal to the new $\hat{x}_k,\hat{y}_k$. The steps required for adding and deleting a key/recollection pair using the above algorithm are summarized in Figs. 7 and 8. The instruction "add $\hat{x}_k$, $\hat{y}_k$ to the memory" means that the outer product $\hat{y}_k\hat{x}_k^T$ should be added to P_3 in Fig. 4 (for the VOP memory), or that $\hat{x}_k$ and $\hat{y}_k$ should be made new columns in P_2 and P_4 of Fig. 2 (for the VIP memory). Likewise, the instruction "delete $\hat{x}_k$, $\hat{y}_k$ from the memory" means that the outer product should be subtracted from the VOP memory, or that the appropriate columns should be removed from the VIP memory.

For updating, this algorithm requires that we store $\mathbf{X}$, $\mathbf{Y}$, $\hat{\mathbf{X}}$, and $\hat{\mathbf{Y}}$. The four matrices contain roughly the same number of elements as the matrices needed for Greville's theorem. Since the gradient projection algorithm is the same for either architecture, the key vector restrictions are the same. A major advantage of the VOP architecture is that its matrix size is much smaller than the VIP architecture's.

The number of multiplications needed in each step using the gradient projection algorithm to add a vector pair to the memory using this algorithm is given in Table IV. Table IV includes the computation of the VOPs, even though this can be performed optically on the architecture of Fig. 4, and is not required at all in the VIP architecture of Fig. 2. These are not major elements

in the computations, and are included for completeness. Under the assumptions $K \ll M < N$, the gradient projection algorithm is the fastest of those discussed in this paper. Its computational load is dominated by the term 2NM (from Table IV). For the next fastest method, bordering, the dominant terms are 2NM and M^2 (from Table 3). Therefore gradient projection is faster than bordering, and is an attractive updating method. To delete a vector pair using the gradient projection algorithm requires between 0 (for the last vector pair) and roughly NM^2 (for the first vector pair). Numerical values for the speed differences between the various algorithms are given in Sec. IX.

IX. Conclusion

Under the realistic assumption (for pattern recognition) that $K \ll M < N$, we have shown that the pseudoinverse heteroassociative memory implemented on the optical matrix-vector multiplier of Fig. 1 has a much smaller number of active elements than the data matrix heteroassociative memory, which must be implemented on the system of Fig. 2. Although the data matrix is trivial to update and has no key vector restrictions, the much smaller size of the pseudoinverse memory and the need for a maximum element selection device at P_3 of the data matrix processor has motivated our search for fast and efficient pseudoinverse updating algorithms. For updating, we have given attention to both adding and deleting vector pairs. Deleting has not been previously considered.

Five selected algorithms have been described and compared with attention to the number of multiplications required, storage, and key vector restrictions. The speed of the different algorithms on the many possible optical linear algebra architectures is another criterion, but is beyond the scope of our present effort. The five algorithms we considered are: bordering, KL, Greville, Widrow-Hoff, and gradient projection. The comparison of these algorithms for associative memories is new and important. In addition, three of the five algorithms are new formulations, optimized for fast updating. The bordering algorithm's speed has been greatly

increased with a new method that partitions the key and recollection matrices. A new bordering method has also been described for deleting vector pairs. The KL algorithm has been adapted for use in an associative memory and its speed has been greatly increased with a new realization that omits unnecessary computations. The gradient projection algorithm has been modified for implementation on VIP and VOP optical processors.

Our comparison results are now summarized with the speed and storage considerations compared quantitatively for a specific example. The qualitative differences found for this example tend to hold true for other values of K, M, and N and the differences for this example agree with our qualitative comparisons at the ends of Secs. V-VIII. We consider an HAM with N = 1000, M = 500, C = 10 and K = 10 and assume linearly independent keys for the sake of comparison. For the KL algorithm, we let $\eta = 6$. The number of floating point multiplications needed to add a vector pair to this AM using direct computation, bordering, KL, Greville, Widrow-Hoff (best case), and gradient projection are 3.2×10^8, 1.3×10^6, 4.1×10^6, 1.5×10^6, 1.0×10^7, and 1.0×10^6 respectively. In Sec. VII, we discussed why the Widrow-Hoff algorithm is slower at adding a vector pair than the other four updating algorithms. We note that the four updating algorithms' computation times are all of the same order of magnitude O(NM) and two orders of magnitude less than direct calculation. The number of multiplications needed to delete a vector pair from this associative memory using direct computation, bordering, KL, Greville, Widrow-Hoff, and gradient projection (average number) are 3.2×10^8, 2.6×10^8, 5.9×10^6, 3.2×10^8, 0, and 1.3×10^8 respectively. Since an efficient updating method does not exist for Greville's algorithm, a vector pair must be deleted by direct computation in this case. The number of matrix elements that must be stored externally for direct computation and the same five algorithms are 5.1×10^5, 6.4×10^5, 6.0×10^5, 5.3×10^5, 1.0×10^6, and 1.0×10^6. We have considered the storage requirement to be the least important criterion for the updating algorithms (the storage requirements only differ by a factor of two for all of the algorithms). From the standpoint of the speed of adding a vector pair, all algorithms are comparable (except

for KL, which is over two times slower, and Widrow-Hoff, which is slower still).

Another comparison criterion is the algorithms' restrictions on the key vectors. The bordering algorithm requires linearly independent keys. For the gradient projection algorithm, if a key vector is linearly dependent, its recollection vector must be the corresponding linear combination of the other recollections. The other three algorithms have no restrictions on the key vectors and are thus preferable.

Each of the five algorithms has an advantage. To select the preferable updating algorithm for a particular application, one must decide on the relative importance of adding vector pairs, deleting vector pairs, and the likelihood of linearly independent key vectors. For pattern recognition applications, we expect adding to be more important than deleting, and we must allow for linearly dependent keys. The need to handle linearly dependent key vectors omits bordering and gradient projection from consideration. From speed considerations, the Greville algorithm is the fastest of the remaining algorithms. Thus, we conclude that Greville's algorithm is preferable.

Acknowledgements

The support of this research by the Air Force Office of Scientific Research (Grant AFOSR-84-0293) and the partial support by the internal research and development funds of General Dynamics (Agreement No. 752647) is gratefully acknowledged.

References

1. J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proc. Natl. Acad. Sci. USA **79,** 2554 (1982).
2. Kwan F. Cheung, Les E. Atlas, and Robert J. Marks II, "Synchronous vs. Asynchronous Behavior of Hopfield's CAM Neural Net," Applied Optics **26,** 4808 (1987).
3. G.R. Gindi, A.F. Gmitro, and K. Parthasarathy, "Hopfield Model Associative Memory with Nonzero-Diagonal Terms in Memory Matrix," Applied Optics (1988).
4. B. Montgomery and B.V.K. Vijaya Kumar, "An Evaluation of the Use of the Hopfield Neural Network Model as a Nearest-Neighbor Algorithm," Applied Optics **25,** 3759 (1986).
5. Teuvo Kohonen, *Self-Organization and Associative Memory, 2nd edition* (Springer-Verlag, Berlin, 1988).
6. C.R. Rao and S.K. Mitra, *Generalized Inverse of Matrices and Its Applications* (John Wiley and Sons, New York, 1971).
7. C.L. Giles and T. Maxwell, "Learning, Invariance, and Generalization in High-Order Neural Networks," Applied Optics **26,** 4972 (1987).
8. D. Psaltis and C.H. Park, "Nonlinear Discriminant Functions and Associative Memories," *Neural Networks for Computing,* J.S. Denker,ed., Am. Inst. of Physics, Snowbird, UT, 1986, p. 370.
9. Ravindra Athale, Harold Szu, and Carl Friedlander, "Optical Implementation of Associative Memory with Controlled Nonlinearity in the Correlation Domain," Optics Letters **11,** 482 (1986).
10. B. Kosko, "Bidirectional Associative Memories," IEEE Trans. Sys. Man and Cybern. **18,** 49 (1988).
11. C.C. Guest and R. TeKolste, "Designs and Devices for Optical Bidirectional Associative Memories," Applied Optics **26,** 5055 (1987).

12. Y. Owechko, "Optoelectronic Resonator Neural Networks," Applied Optics **26,** 5104 (1987).

13. D.Z. Anderson and M.C. Erie, "Resonator Memories and Optical Novelty Filters," Optical Engineering **26,** 434 (1987).

14. David Casasent and Brian Telfer, "Distortion-Invariant Associative Memories and Processors," *Proc. SPIE, Vol. 697,* Aug. 1986, p. 60.

15. David Casasent, "Scene Analysis Research: Optical Pattern Recognition and Artificial Intelligence," *Optical and Hybrid Computing,* Proc. SPIE, vol. 634, Mar. 1986, p. 439.

16. B. Kosko, "Adaptive Bidirectional Associative Memories," Applied Optics **26,** 4947 (1987).

17. Nabil Farhat, "Optoelectronic Analogs of Self-Programming Neural Nets: Architecture and Methodologies for Implementing Fast Stochastic Learning by Simulated Annealing," Applied Optics **26,** 5093 (1987).

18. R.A. Athale and W.C. Collins, "Optical Matrix-Matrix Multiplier Based on Outer Product Decomposition," Applied Optics **21,** 2089 (1982).

19. A.D. Fisher, W.L. Lippincott, and J.N. Lee, "Optical Implementations of Associative Networks with Versatile Adaptive Learning Capabilities," Applied Optics **26,** 5039 (1987).

20. T. Daud et. al., "Feedforward, High Density, Programmable Read Only Neural Network Based Memory System," *High Speed Computing,* D. Casasent,ed., Proc. SPIE, vol. 880, 1988, p. 76.

21. Richard P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine **4,** 4 (1987).

22. G. Gindi, A. Gmitro, and K. Parthasarathy, "Winner-Take-All Networks and Associative Memory: Analysis and Optical Realization," *Proc. International Conf. on Neural Networks, Vol. III,* IEEE, San Diego, June 1987, p. 607

23. D.K. Faddeev, *Computational Methods of Linear Algebra* (W.H. Freeman & Co., London, 1963).

24. Joan Westlake, *A Handbook of Numerical Matrix Inversion and Solution of Linear Equations* (John Wiley & Sons, New York, 1968).

25. B.H. Soffer et. al., "Programmable Real-Time Incoherent Matrix Multiplier for Optical Processing," Applied Optics **25**, 2295 (1986).

26. David Casasent and Wen-Thong Chang, "Correlation Synthetic Discriminant Functions," Applied Optics **25**, 2343 (1986).

27. K. Fukunaga, *Introduction to Statistical Pattern Recognition* (Academic Press, New York, 1972).

28. Hiroyasu Murakami and B.V.K. Vijaya Kumar, "Efficient Calculation of Primary Images from a Set of Images," IEEE Trans. Pat. Anal. and Mach. Int. **PAMI-4**, 511 (1982).

29. Gilbert Strang, *Linear Algebra and Its Applications* (Harcourt Brace Jovanovich, Orlando, 1980).

List of Figures

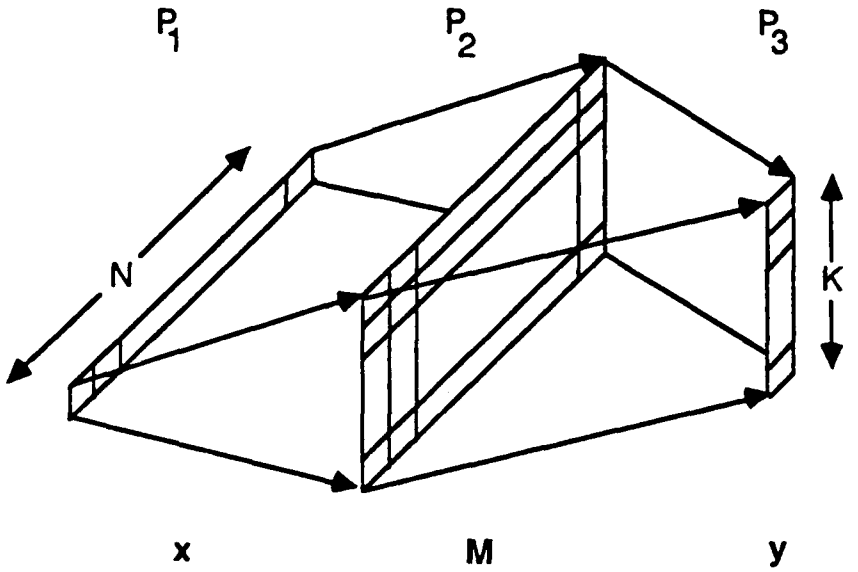


Figure 1: Analog optical matrix-vector multiplier for pseudoinverse associative memory

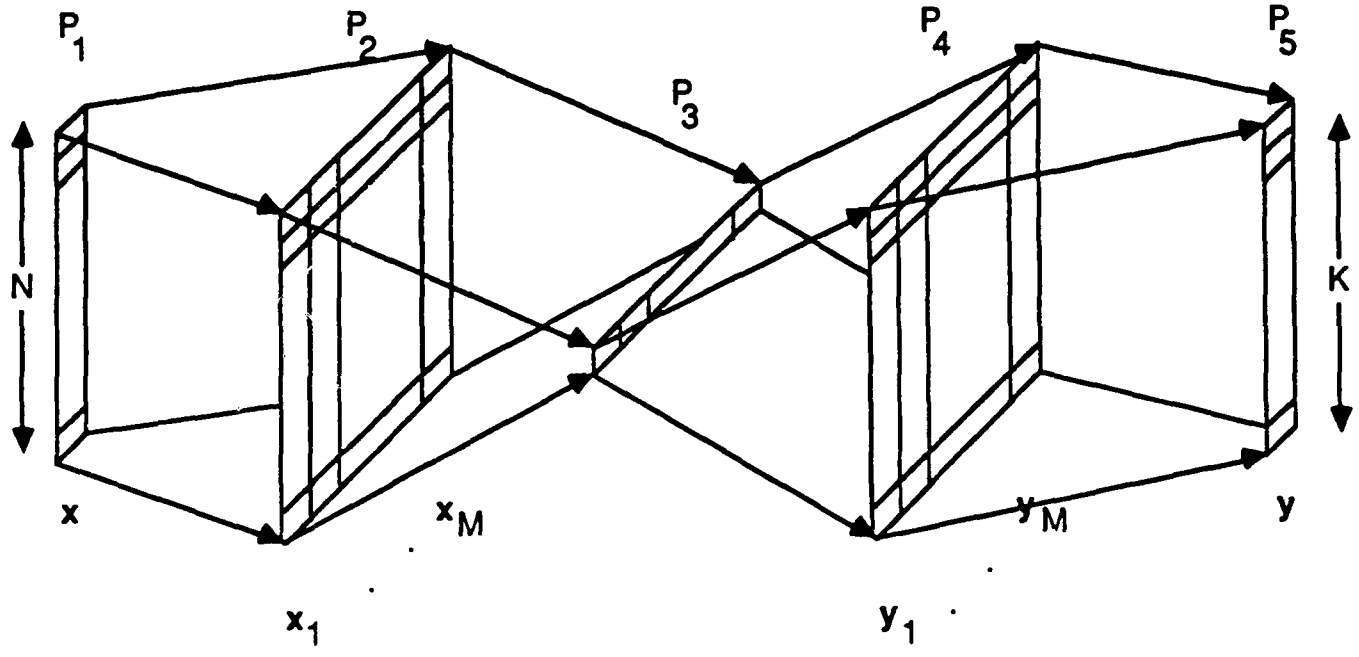


Figure 2: Vector inner product optical associative memory architecture

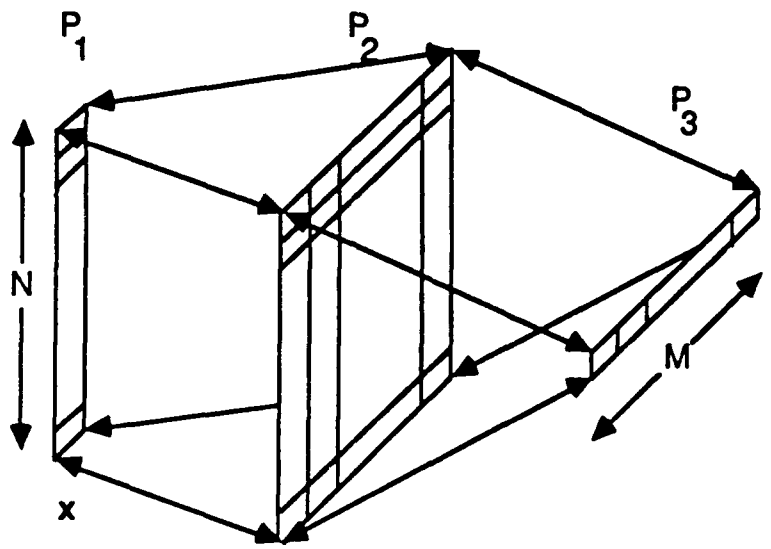


Figure 3: Optical bidirectional vector inner product autoassociative memory architecture

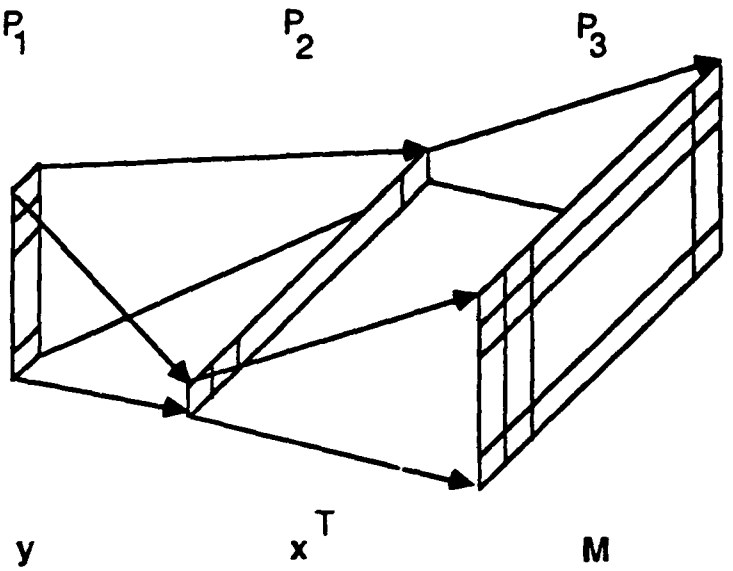


Figure 4: Vector outer product architecture for forming an associative memory matrix

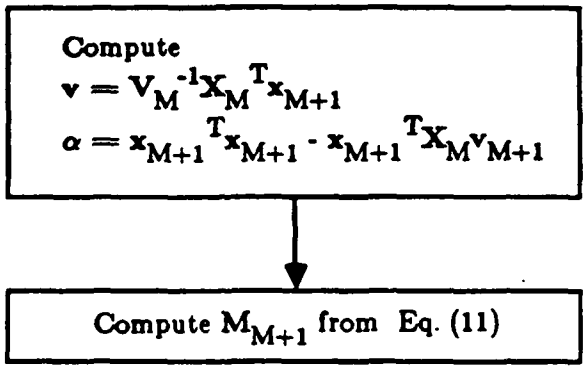


Figure 5: Flowchart for adding a vector pair to an associative memory using the bordering algorithm

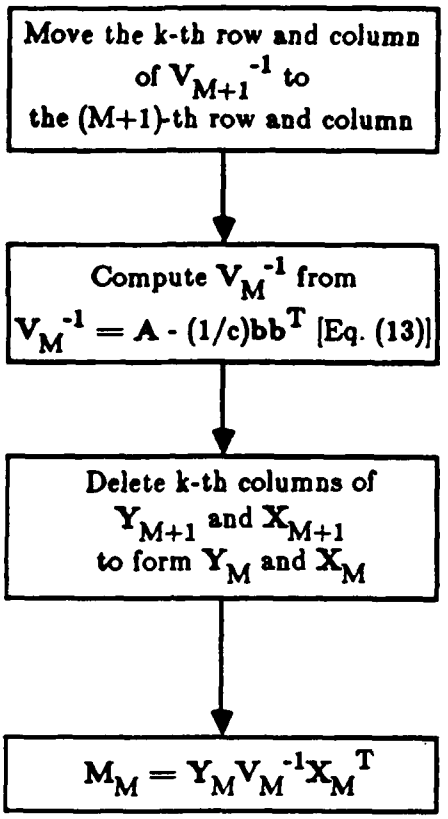


Figure 6: Flowchart for deleting a vector pair from an associative memory using the bordering algorithm

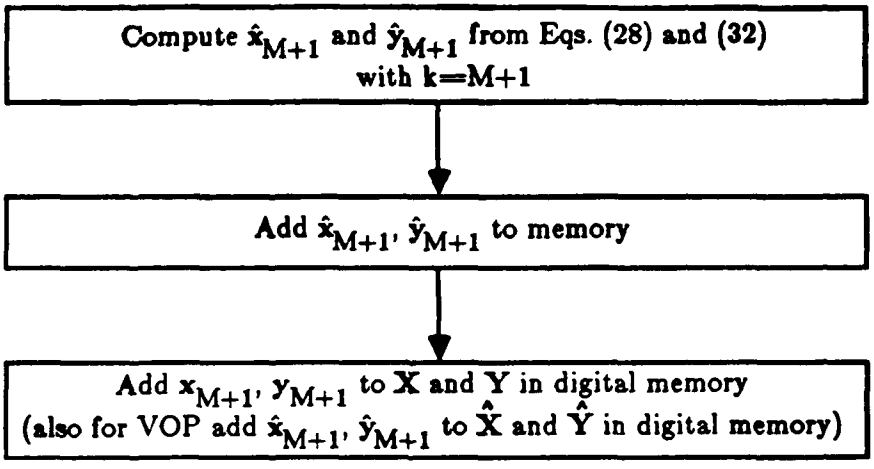


Figure 7: Flowchart for adding a vector pair to an associative memory using the gradient projection method

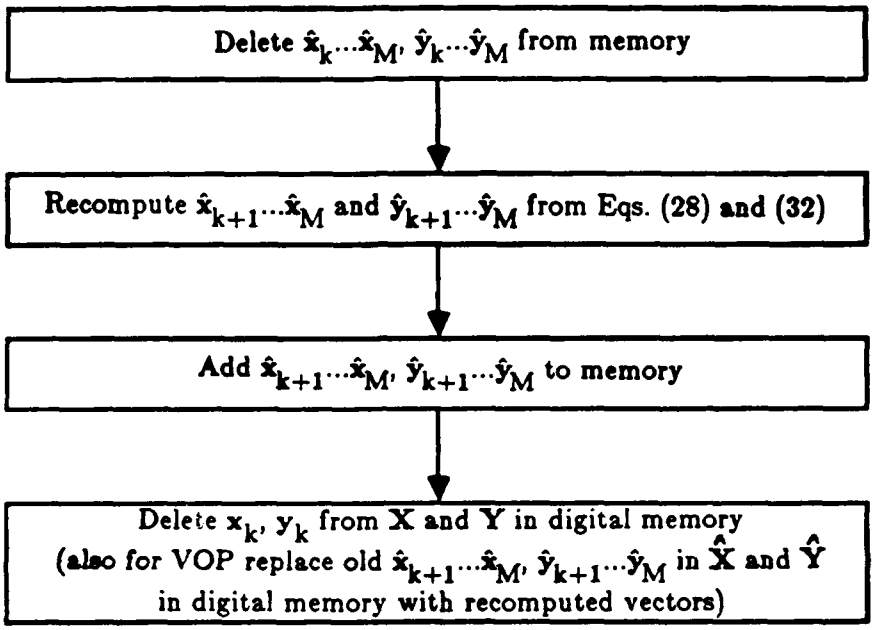


Figure 8: Flowchart for deleting x_k, y_k from an associative memory using the gradient projection method

List of Tables

Operation	No. Multiplications
$X_M^T x_{M+1}$	NM *
$v = V_M^{-1}(X_M^T x_{M+1})$	M^2 *
$X_M v$	NM *
$(1/\alpha)Y_M v$	KM
$(1/\alpha)Y_M v(v^T X_M^T)$	KN
$[(1/\alpha)Y_M v]x_{M+1}^T$	KN
$(1/\alpha)y_{M+1}(v^T X_M^T)$	KN
$(1/\alpha)y_{M+1}x_{M+1}^T$	KN
TOTAL	$2NM + M^2 + 4KN + KM$
	$\approx 2NM + M^2$ (for $K \ll M < N$)

Table I: Number of multiplications required for adding a vector pair to an associative memory using our improved bordering algorithm

(* indicates the dominant operations when $K \ll M < N$)

Operation	No. Multiplications
$\mathbf{A} = \tilde{\mathbf{X}}^T\mathbf{X}$ [Eq. (15)]	ηNM *
$\mathbf{E} = \mathbf{Y}\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}$ [Eqs. (17) and (19)]	$(\eta C + K)\eta M$
$\mathbf{M} = \mathbf{E}\tilde{\mathbf{X}}^T$ [Eq. (19)]	ηCKN *
TOTAL	$\eta NM + \eta CKN + (\eta C + K)\eta M$
	$\approx \eta NM + \eta CKN$ (for $K \ll M < N$)

Table II: Number of multiplications required for adding a vector pair to an associative memory using the KL algorithm, assuming small η

(* indicates the dominant operations when $K \ll M < N$)

Operation	No. Multiplications
$X_M^+ x_{M+1}$	NM *
$X_M(X_M^+ x_{M+1})$	NM *
$(X_M^+)^T(X_M^+ x_{M+1})$	NM * for linearly dependent x_{M+1}
$(X_M^+ x_{M+1})\rho^T$	NM *
$M_M x_{M+1}$	KN
$(M_M x_{M+1})\rho^T$	KN
TOTAL	3NM + 2KN for linearly independent x_{M+1} ≈ 3NM (for K<<M<N) 4NM + 2KN for linearly dependent x_{M+1} ≈ 4NM (for K<<M<N)

Table III: Number of multiplications required for adding a vector pair to an associative memory using Greville's Theorem

(* indicates the dominant operations when $K \ll M < N$)

Operation	No. Multiplications
$\hat{x}_{M+1}^T \hat{x}_i$	NM *
$(\hat{x}_{M+1}^T \hat{x}_i)\hat{x}_i$	NM *
$(\hat{x}_{M+1}^T \hat{x}_i)\hat{y}_i$	KM
$\hat{y}_{M+1} \hat{x}_{M+1}^T$	KN
TOTAL	2NM + KN + KM ≈ 2NM (for K<<M<N)

Table IV: Number of multiplications required for adding a vector pair to an associative memory using the gradient projection method

(* indicates the dominant operations when $K \ll M < N$)

CHAPTER 9:

"OPTICAL ASSOCIATIVE PROCESSORS FOR VISUAL PERCEPTION"

Proc. SPIE
Vol. 882
January 1988

"OPTICAL ASSOCIATIVE PROCESSORS FOR VISUAL PERCEPTION"

David Casasent and Brian Telfer
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

ABSTRACT

We consider various associative processor modifications required to allow these systems to be used for visual perception, scene analysis, and object recognition. For these applications, decisions on the class of the objects present in the input image are required and thus heteroassociative memories are necessary (rather than the autoassociative memories that have been given most attention). We analyze the performance of both associative processors and note that there is considerable difference between heteroassociative and autoassociative memories. We describe associative processors suitable for realizing functions such as: distortion invariance (using linear discriminant function memory synthesis techniques), noise and image processing performance (using autoassociative memories in cascade with with a heteroassociative processor and with a finite number of autoassociative memory iterations employed), shift invariance (achieved through the use of associative processors operating on feature space data), and the analysis of multiple objects in high noise (which is achieved using associative processing of the output from symbolic correlators). We detail and provide initial demonstrations of the use of associative processors operating on iconic, feature space and symbolic data, as well as adaptive associative processors.

1. INTRODUCTION

Optical neural networks offer the promise of adaptive pattern recognition. In this paper, we discuss optical associative processors and their role in scene analysis and visual perception. Specific issues arise in this problem that require new associative processors that we will address:

1. We require classification of objects in the scene and thus require heteroassociative memories (HAMs) rather than the autoassociative memories (AAMs) discussed in the majority of the literature [1]. In Section 2, we summarize results [2,3] that indicate how the performance of HAM architectures significantly differ from that of AAMs.

2. For adaptive associative processors, we require techniques to update (add, delete, and substitute) new key and recollection vector pairs. This is discussed and detailed in Section 3.

3. Distortion-invariance is needed for object recognition. To achieve this, we incorporate linear discriminant function (LDF) synthesis techniques into the formation of associative processors [2]. These associative processors will be shown to have significantly reduced size and storage requirements (Section 4).

4. Operation in noise and with partial occluded inputs is also required. We achieve this with AAM/HAM cascades (Section 5) and with symbolic correlators (Section 7).

5. Shift invariance is also required for image analysis (i.e., the location of the object in the field of view must not affect performance). We achieve this by a feature space image representation key input to an associative processor (Section 6).

6. Multiple objects and objects in clutter must also be considered. We address this by a symbolic correlator with an output associative processor operating on symbolic word object descriptions (Section 7).

This paper denotes several new associative processors operating on iconic, feature space and symbolic data, as well as adaptive processors and the use of such systems for scene analysis and visual perception.

We denote the input key vectors to our optical associative processor by $\{\underline{x}_k\}$. These vectors are of dimension N. The output recollection vectors $\{\underline{y}_k\}$ are of dimension K and the memory matrix $\underline{M}$ is thus K×N. Figure 1 shows one simple realization of such an associative processor. Holographic, phase conjugation, and other techniques can also be employed to realize such processors. The storage capacity of such a processor is given by the number of key/recollection vector pairs (M) that the system can accommodate. We group the different key and recollection vectors into columns and form key and recollection matrices $\underline{X}$ and $\underline{Y}$ respectively. The matrix $\underline{M}$ must satisfy $\underline{M}\,\underline{x}_k = \underline{y}_k$ for all $\{\underline{x}_k\}$ and $\{\underline{y}_k\}$. Thus, $\underline{M}$ must solve $\underline{M}\,\underline{X} = \underline{Y}$. If $\underline{x}_k = \underline{y}_k$, then $\underline{X} = \underline{Y}$ and an autoassociative memory results. These memories are useful to reduce noise in the input $\underline{x}_k$ data and to restore missing parts in the input data. If $\underline{y}_k$ is an encoded bit pattern (such as a unit vector or a binary-encoded word), then a heteroassociative memory results and the output recollection vector code $\{\underline{y}_k\}$ denotes the object class, orientation, etc. Examples of these systems are provided in Section 4.

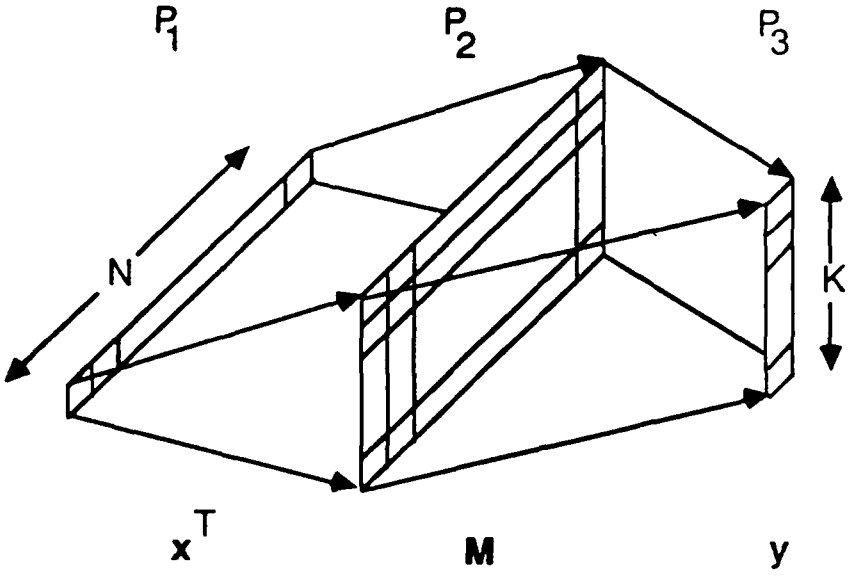


FIGURE 1:
Basic Optical Matrix-Vector Realization of an Associative Processor

2. HAM PERFORMANCE

We recently [2] detailed that the conventional AAM performance measure (output to input noise variance σ_o^2/σ_i^2) is a useful measure for an AAM, but is not suitable for HAM performance. We showed that the recollection vector choice affects this performance measure and that HAM performance is not simply $\sigma_o^2/\sigma_i^2 = M/N$, where N is the dimension of the input key vectors and M is the number of key/recollection vector pairs stored. We also noted that different choices for the recollection vector can and do affect this performance measure for the case of HAMs. We derived analytic expressions for σ_o^2/σ_i^2 for AAMs and for several types of HAMs. We have also noted and quantified AAM and HAM performance for different types of HAMs (with different recollection vector encodings). These tests indicated that HAM performance requires a new performance measure. We introduced [2] the output to input SNR (SNR_o/SNR_i) as such a measure, whose performance cannot be arbitrarily scaled to artificially increase results, as is the case when σ_o^2/σ_i^2 is used. In this recent work, we quantified HAM SNR performance (i.e. the error correction ability of the associative memory itself), we then related P_c (the probability of correct recognition) to the output noise standard deviation σ_o (i.e. the noise tolerance of the recollection vector encoding). We quantified and showed that the SNR ratio performance of an HAM with binary encoded rather than unit recollection vectors was preferable, that conventionally encoded HAMs need maximum element detectors, and that binary encoded output recollection vectors yield proportionally better performance (i.e. less loss in noise reduction) than do unit recollection vectors as M increases. The reduced size of the HAM memory that results when such recollection vector encodings are used is another motivation for their use.

3. ASSOCIATIVE MEMORY UPDATING FOR ADAPTIVE PROCESSING

Adaptive processing requires updating of the associative memory (adding new key/recollection pairs deleting old key/recollection pairs, or substituting for a prior vector pair). The techniques used to achieve this depend on how the associative memory is synthesized. The matrix $\underline{M}$ must satisfy $\underline{M}\,\underline{X} = \underline{Y}$. For the pseudoinverse solution $\underline{M} = \underline{Y}\,\underline{X}^+$, where $\underline{X}^+ = (\underline{X}^T\underline{X})^{-1}\underline{X}^T$, and where $\underline{X}^T\underline{X} = \underline{V}$ is the vector inner product (VIP) matrix. For the case of M key/recollection pairs, we find

$$\underline{M}_M = \underline{Y}_M\underline{V}_M^{-1}\underline{X}_M^T, \tag{1}$$

where the subscripts indicate the number of stored key/recollection pairs. If the key vectors are orthonormal, then $\underline{V}_M = \underline{I}$, and $\underline{X}^+ = \underline{X}^T$ and the matrix calculation simplifies to

$$\underline{M}_M = \underline{Y}_M\underline{X}_M^T. \tag{2}$$

If we use the matrix-vector architecture of Figure 1 with $\underline{M}$ given by Eq.(1), then we require linearly independent key vectors (this generally occurs when the input data are iconic image representations that are significantly different; however, this property is not necessarily always satisfied). We update $\underline{M}_M$ to

$\underline{M}_{M+1}$ by adding new $\underline{x}_{M+1}$ and $\underline{y}_{M+1}$ vectors as columns to $\underline{X}_M$ and $\underline{Y}_M$ and by updating $\underline{V}_M^{-1}$ to $\underline{V}_{M+1}^{-1}$ by the bordering algorithm

$$\underline{V}_{M+1}^{-1} = \begin{bmatrix} \underline{V}_M^{-1}+(1/a)\underline{v}_{M+1}\underline{v}_{M+1}^T & -(1/a)\underline{v}_{M+1} \\ -(1/a)\underline{v}_{M+1}^T & (1/a) \end{bmatrix}, \tag{3}$$

where $\underline{v}_{M+1} = \underline{V}_M^{-1}\underline{X}_M^T\underline{x}_{M+1}$ and $a = \underline{x}_{M+1}^T\underline{x}_{M+1} - \underline{x}_{M+1}^T\underline{X}_M\underline{v}_{M+1} =$ scalar. This calculation of the inverse of the new $\underline{V}$ from the inverse of the old $\underline{V}$ is much simpler than computing the inverse of the new $\underline{V}$ matrix. To delete the k-th vector pair from $\underline{V}_{M+1}^{-1}$ to produce $\underline{V}_M^{-1}$, we move the k-th row and column of $\underline{V}_{M+1}^{-1}$ to the last row and column. This yields a partitioned matrix

$$\underline{V}_{M+1}^{-1} = \begin{bmatrix} \underline{A} & \underline{b} \\ \underline{b}^T & (1/a) \end{bmatrix}. \tag{4}$$

Comparing Eqs.(3) and (4), we find that $\underline{V}_M^{-1}$ is given by the new result

$$\underline{V}_M^{-1} = \underline{A} - a\ \underline{b}\,\underline{b}^T. \tag{5}$$

With orthonormal key vectors, the vector outer product (VOP) realization of an associative memory follows, since $\underline{M} = \underline{Y}\,\underline{X}^T = \Sigma\underline{y}_k\underline{x}_k^T = \Sigma$ VOPs. The VOP optical architecture is shown in Figure 2. A VIP architecture can be realized for an input key $\underline{x}$ to yield an output recollection vector $\underline{y}$, since $\underline{M}\,\underline{x}$ $\underline{Y}\,\underline{X}^T\underline{x} = \underline{y}$. The VIP architecture [5] is shown in Figure 3.

With a nonlinear operation performed at P_3 of Figure 3, orthonormal key vectors are not required in the VIP architecture. If the P_3 nonlinearity is a maximum selector, the P_1 to P_3 portion of Figure 3 is a data matrix associative processor. Such an architecture has been shown [6] to be a preferable neural network processor to the more widely known and used Hopfield memory.

Orthogonal key vectors $\hat{\underline{x}}_k$ are calculated from the $\underline{x}_k$ by Gram-Schmidt techniques and the associated $\hat{\underline{y}}_k$ recollection vectors are similarly determined from the $\underline{y}_k$,$\underline{x}_k$ and $\hat{\underline{x}}_k$. These Gram-Schmidt steps required on each new input vector can be achieved on-line using dedicated hardware. To update or add a new $(\underline{x}_{M+1},\underline{y}_{M+1})$ vector pair to the system, we compute the new portion $\hat{\underline{x}}_{M+1}$ of $\underline{x}_{M+1}$ and the associated $\hat{\underline{y}}_{M+1}$. We add this to the VOP system by adding the VOP $\hat{\underline{x}}_{M+1}\hat{\underline{y}}_{M+1}^T$ to the memory matrix. For the VIP system, we add $\hat{\underline{x}}_{M+1}$ and $\hat{\underline{y}}_{M+1}$ as columns to $\underline{X}_M$ and $\underline{Y}_M$. To delete the vector pair $(\underline{x}_k,\underline{y}_k)$, we must remove all vector pairs k, k+1, etc. and recalculate all vector pairs k+1 etc. This is necessary, since the subsequent key vectors after the k-th vector no longer need to be orthogonal to the k-th key vector. To substitute a new $(\underline{x}_k,\underline{y}_k)$ pair for an old pair, we delete the old pair (and subsequent pairs), we then calculate the new $(\hat{\underline{x}}_k,\hat{\underline{x}}_k)$ pair and all subsequent pairs and add

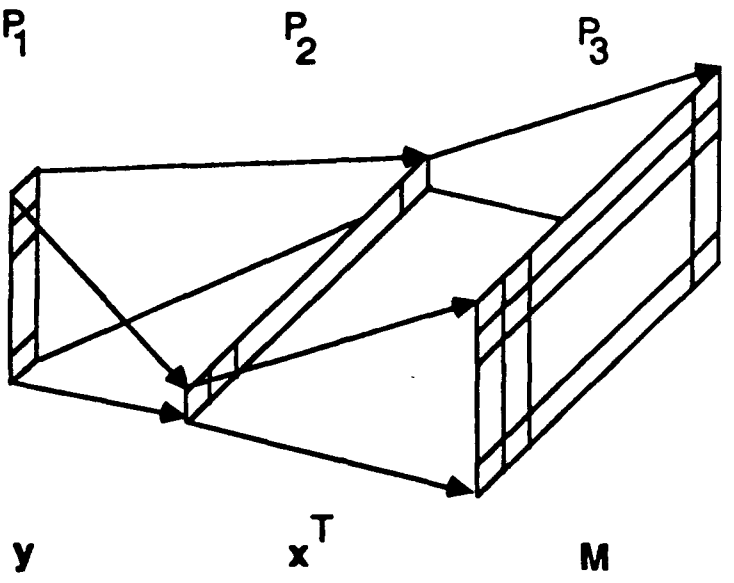


FIGURE 2:
Vector Outer Product Optical Associative Memory Architecture

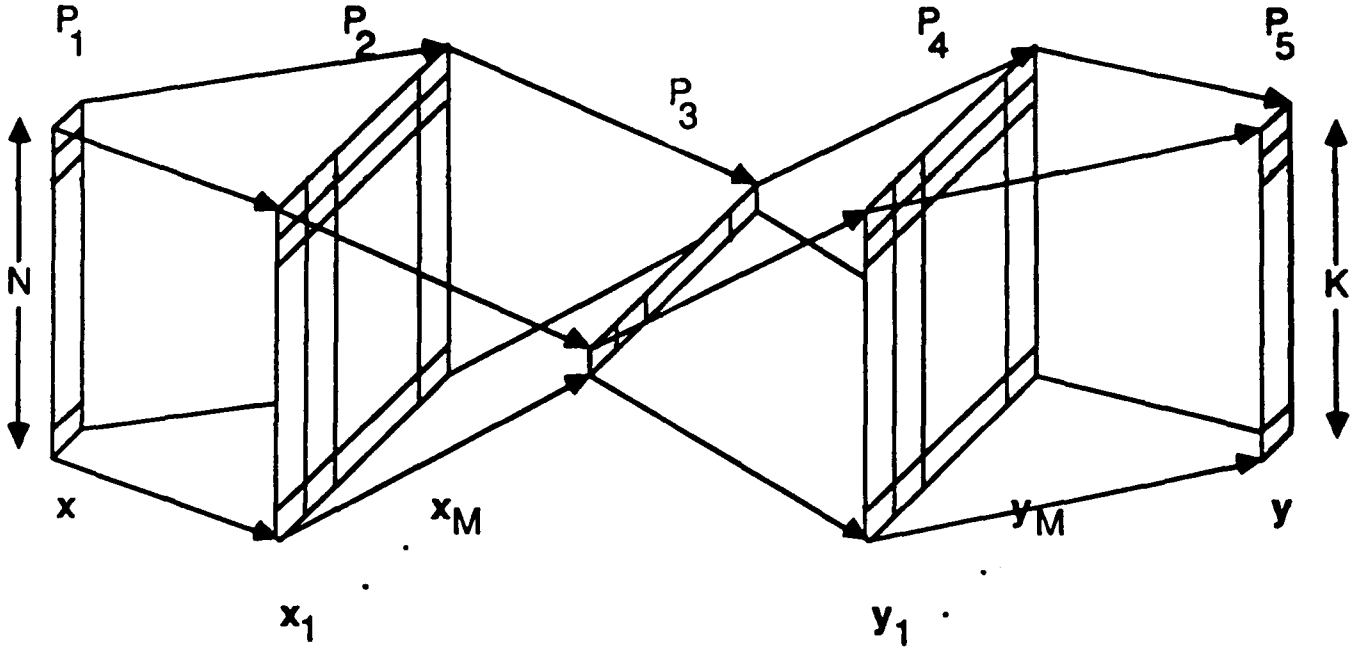


FIGURE 3:
Vector Inner Product Optical Associative Memory Architecture

these new pairs to the matrix.

4. DISTORTION-INVARIANT ASSOCIATIVE PROCESSORS AND LDF ASSOCIATIVE PROCESSOR SYNTHESIS

In vision processing, we require distortion-invariant object recognition (or at least some limited

amount of distortion-invariance). We achieve this by using LDFs in the synthesis of the memory matrix [4]. Specifically, for a two-class problem with in-plane rotations as the distortions to be handled, the matrix used is only $2\times d$ (where the number of pixels on the object is d) and the two element output recollection vector is $[1,0]^T$ or $[0,1]^T$ to denote the two classes independent of object distortions. The input key vector is $d\times 1$ (lexicographically ordered iconic or pixel-based image). In synthesis of this memory matrix, we specify the same output recollection vectors $[1,0]^T$ or $[0,1]^T$ for different distorted input key vectors. The discriminant vectors are linear combinations of distorted versions of the two objects. Figure 4 shows an example of such an associative processor. We used 18 Phantom and 18 DC-10 aircraft images, 32×32 pixels, rotated in yaw every 20°. From these, two discriminant vectors were formed and the 2×1024 memory was produced. Figure 4 shows the Phantom and DC-10 test images at several different orientations (the system has only seen the 0° image before and none of the other test images), and the two element recollection vector obtained. As seen, the system correctly classifies the Phantom inputs with a $[1,0]^T$ output and DC-10 inputs with a $[0,1]^T$ output.

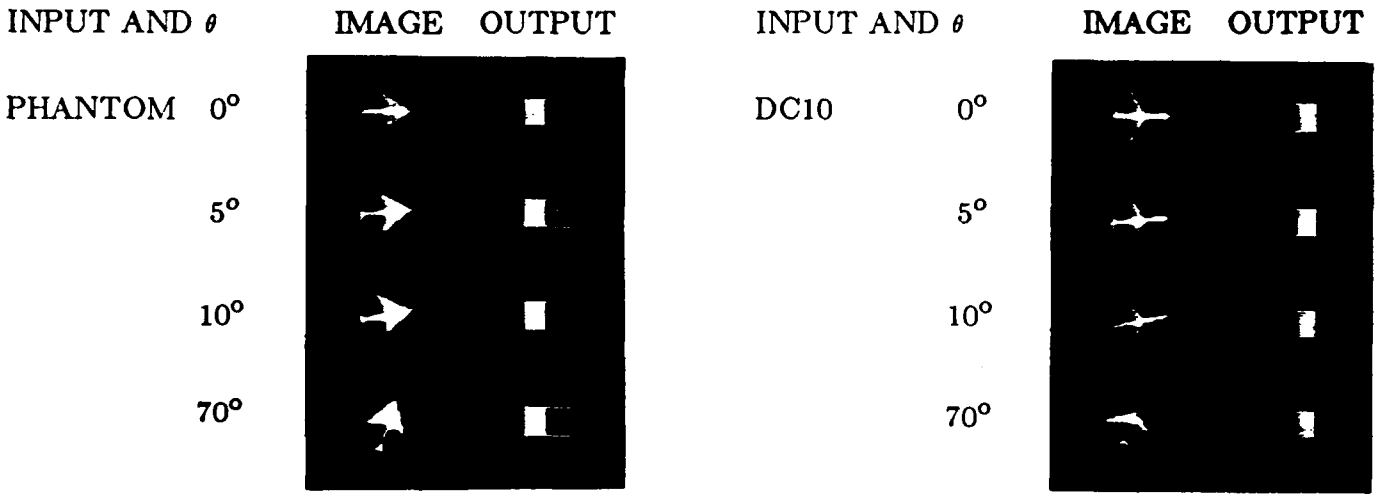


FIGURE 4:
Demonstration of In-Plane Distortion-Invariant Associative Processing

In many cases, the class of the input object is sufficient (as in Figure 4) for a given problem. After such an analysis, further scene interpretation often requires determination of the orientation of the object. This is also necessary in robotic applications, when the object must be grasped. We achieve designation of the class of the input object by a different associative processor. Instead of assigning the same key vectors to all distorted versions of one input object, we can assign different unit recollection vectors to the different object orientations, i.e. the location of the single "1" in the output vector denotes the class of the object. Figure 5 shows the resultant object orientation estimates obtained from such an associative processor. In this case, 10 LDF vectors were formed (the 10 rows of the matrix $\underline{M}$) as linear combinations of 10 images of the Phantom aircraft at 36° increments in yaw. The input vectors are 1024×1, the memory matrix is 10×1024, and the 10×1 recollection vectors were the 10 different unit vectors (for the 10 different yaw rotated key images). Figure 5 shows the input image, its orientation, and the output recollection vector obtained in tests. The system has not previously seen any views between 0° and 36°. The output vectors obtained for the 4°, 8°, 12° and 16° rotated input images have a "1" in the first location (indicating that these images are most closely associated

with the 0° oriented Phantom). The inputs from 20° to 32° gave output recollection vectors with a "1" in the second element (since they are closest to the 36° oriented Phantom), etc.

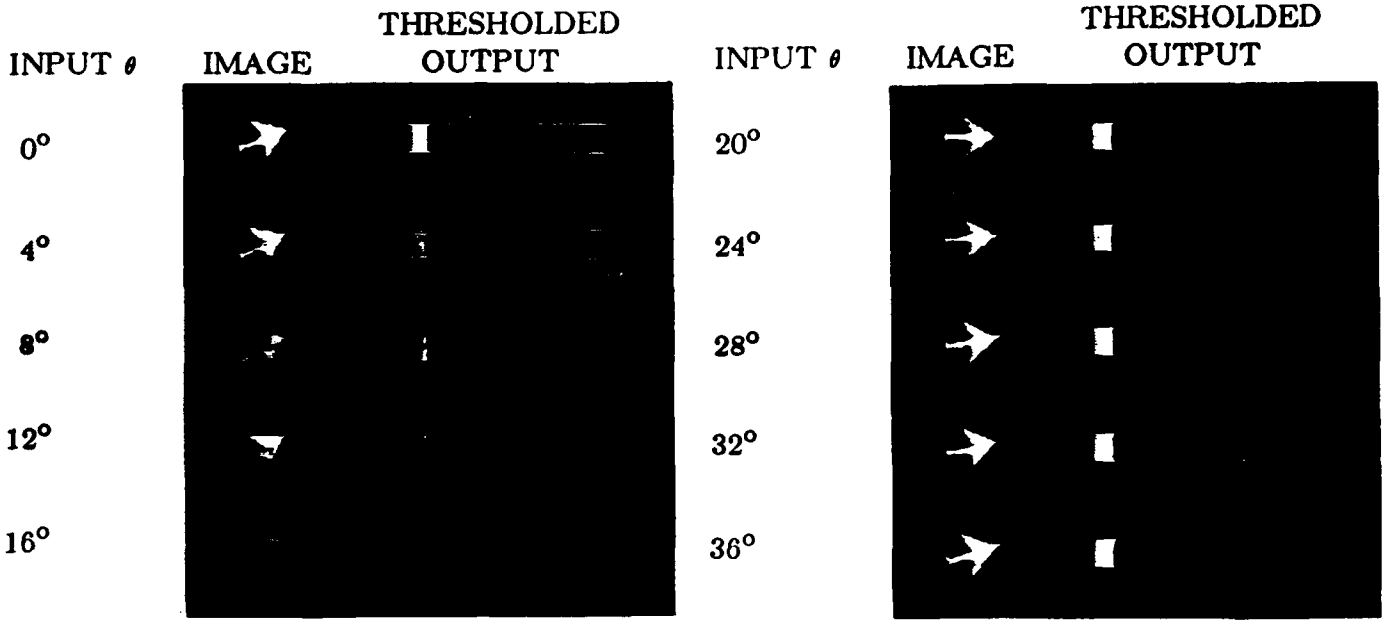


FIGURE 5:
Object Orientation Determination from an Associative Processor

5. ASSOCIATIVE PROCESSORS WITH NOISY INPUTS AND OCCLUDED OBJECTS

When input noise is present or when only apart of the input object is visible (due to occlusion, etc.), HAM performance will degrade. In such cases, AAMs are of use. Their output recollection vectors are of the same type as the input key vector, i.e. iconic image patterns (of improved quality) . Figure 6 shows the inputs and outputs for an associative memory (1024×1024) for input objects with parts missing (Figures 6a to 6d) and with significant noise present (Figures 6e and 6f). With severe noise and occlusion, several iterations through an AAM are found to improve results. This is analogous to neural processors, whose outputs are thresholded and fed back to the input. We have found it useful and necessary to restrict the number of iterations to typically 2 or 3. Following this AAM preprocessing, the resultant input vector is significantly improved (as in Figure 6) and is then fed to an HAM for a decision on the class, orientation, etc. of the object as shown in the block diagram of Figure 7.

6. SHIFT INVARIANT ASSOCIATIVE PROCESSORS WITH INPUT FEATURE VECTORS AS KEY VECTORS

For image and scene analysis, one cannot rely on the fact that the object to be classified will be in the center of the field of view. Thus, associative processors must be able to accommodate objects in any position (i.e. a shift invariant processor is required). With conventional associative processors operating on iconic data, we would require the memory to be synthesized for all possible different shifted locations of the object. This will significantly affect memory storage capacity and dynamic range requirements.

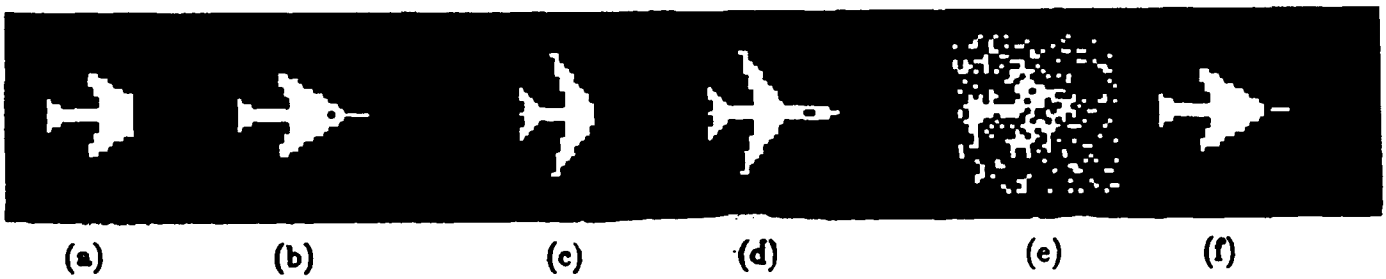


FIGURE 6:
Autoassociative Memory Image Recollection. Partial Phantom Input (a) and Output Obtained (b). Partial DC10 Input (c) and Output Obtained (d). Noisy Phantom Input (e) and Output Obtained (f).

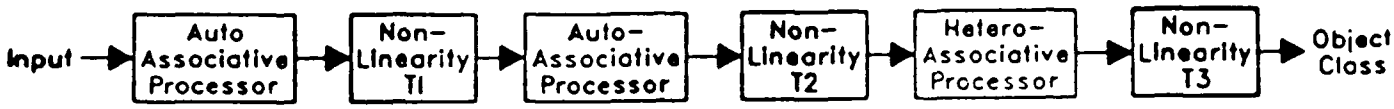


FIGURE 7:
Three-Stage Nonlinear Associative Processor with More Improved Distortion-Invariant Pattern Recognition

Alternatively, one can employ a correlator architecture which automatically checks the input for every possible shifted location of an object. These correlation filters can be formed from smart filters to achieve distortion invariance. If the output from a correlation filter is thresholded and the filter is readout then some reconstruction of objects with occluded parts is possible. However, before achieving this, the class of the object must typically be assessed (and this is often the major problem). With phase conjugate mirror readout, better associative performance is possible. In general, a correlation smart filter that achieves object recognition does not function well in producing restored noise-free object images.

To achieve shift invariance, we use a feature space object description as the input key to the associative processor. The specific feature space we chose is the well known wedge-ring sampled Fourier transform feature space [7]. It is translation invariant (as well as scale and rotation invariant). It is also significantly reduced in dimensionality. The wedge outputs are invariant to scale changes in the input object and the wedge feature vector pattern rotates cyclically as the input object rotates. The magnitude Fourier transform detected provides the shift invariance. We used only wedge-sampled magnitude Fourier transform elements as our feature vector with 32 wedge samples in 180° (i.e. half of the Fourier transform plane is chosen).

Thus, our input key vectors are of dimension 32. As our associative memory matrix, we use the data matrix. We form the matrix from the wedge Fourier transform samples of 9 training images of Phantom jets and 9 of the DC-10 aircraft with the aircraft in different in-plane yaw rotations (specifically 20° intervals in yaw over 180°). Since the magnitude Fourier transform is symmetric, we need only consider 180° coverage or half of the Fourier transform plane. The 32 element input vector is significantly smaller than the 1024 element input vector required for even a small 32×32 resolution input object. The associative memory matrix is likewise much smaller (18×32 in this case). The output recollection vector is of dimension 18, with a single "1" expected. If the single "1" occurs in one

of the first 9 elements, this indicates a Phantom jet input at different yaw orientations at 20° intervals. If the "1" output occurs in one of the last 9 elements of the recollection vector, this indicates a DC-10 input at different yaw orientations at 20° intervals.

Figure 8 shows the Fourier transform of a Phantom jet at 0° yaw orientation. The FT pattern has lines at different angles. These correspond to the different lines in the aircraft image with the length (energy) of each Fourier transform line proportional to the length of the wing etc. lines in the input object. The orientation of these Fourier transform lines is normal to the orientation of the wing etc. lines in the input image. Figure 9 shows the DC-10 image at 0° orientation.

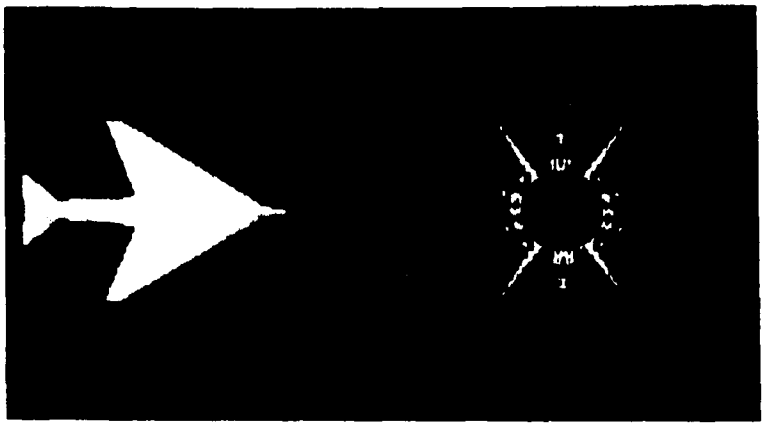


FIGURE 8:
(a) Phantom Jet at 0° and
(b) its Fourier Transform

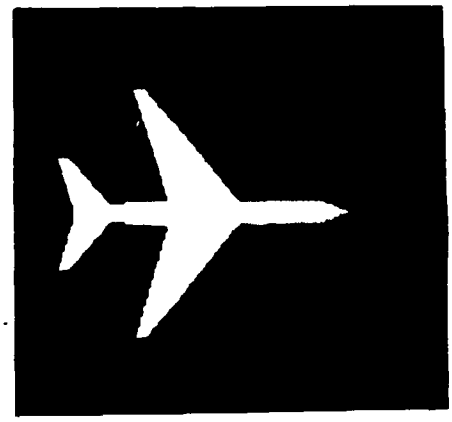
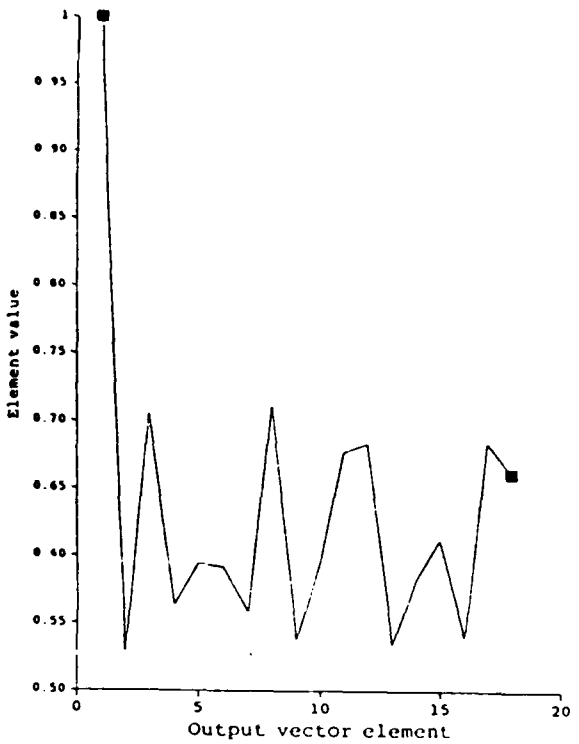


FIGURE 9:
DC-10 aircraft at 0°
orientation

We now show various associative processor data for input feature key vectors. In Figure 10, the input was a Phantom at 0°. The 18 analog recollection vector outputs from the associative processor are shown in Figure 10a and the thresholded 18 element output vector is shown in Figure 10b. The peak value of 1.0 occurs in the first element of the output recollection vector, indicating that the input is a Phantom jet oriented at 0°. For a DC-10 input at 0°, Figure 11 shows analogous data. The peak in the tenth output recollection vector element indicates that the input is a DC-10 at an orientation of 0°. Figures 12 and 13 show test results for non-training set images. In Figure 12, the input was a Phantom at 10°. The peak in the third output element correctly classifies the input as a Phantom, although it does not indicate the correct orientation. In Figure 13, the input was a DC-10 at 10°. Again the output peak identifies the correct class, but not the orientation.

7. ASSOCIATIVE PROCESSORS FOR MULTIPLE OBJECTS IN HIGH CLUTTER (SYMBOLIC KEY INPUT VECTORS)

To develop associative processors suitable for recognizing multiple objects in high clutter with occluded objects and with the ability to update new object models requires advanced choices for the input key vector. To achieve this, we employ a correlator with multiple filters for the object parts in different pattern locations. A correlator with three frequency-multiplexed filters is used. Each matched spatial filter is designed to output a specific 4×4 pattern for different input objects. The location of this 4×4 pattern in each output correlation plane indicates the location of an object in the input field



(a) 18 element analog recollection vector

(b) Thresholded 18 element recollection vector

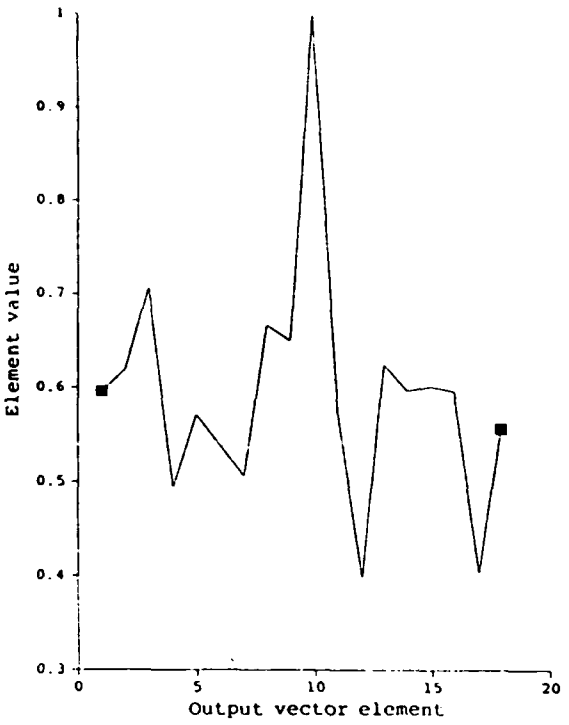
FIGURE 10:
Feature Space Key Vectors Associatively Processed for a Phantom at 10° as the Input.

of view and the specific 4×4 pattern produced indicates the class of the input object. From the same locations in the different output correlation planes, we thus obtain three words (16 digit symbolic words) and from the full output we obtain a three word 48 digit symbolic description of the input object. For multiple objects, three different symbolic words occur (these are separated for the different objects by the correlator). The initial rule-based processor [8] used for this system was not able to handle input objects with occluded parts with high confidence. Figure 14 shows the results of the output symbolic data from such a system when the input had missing occluded object parts. The initial output symbolic data had 13 errors. After processing by an AAM, the number of errors was reduced to 9 and the rule-based processor properly classified the input object with improved confidence.

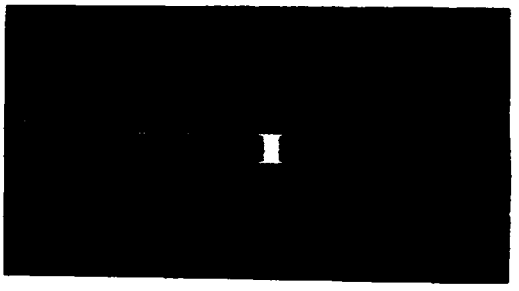
For the case when the input object cannot be classified with high confidence (i.e. it is a new type of object) even with the AAM image improvement, we update our model with this input or with parts of it. To estimate the object class of this new input object, we feed the three symbolic output words describing it to a HAM and perform a majority vote on the outputs. Figure 15 shows this concept. For the case given, this resulted in the proper classification of the input object.

8. SUMMARY AND CONCLUSION

We have addressed associative processors for vision and object recognition. We noted the need for heteroassociative processors for making decisions. Autoassociative memories, cascades of AAMs, as well



(a) 18 element analog recollection vector



(b) Thresholded 18 element recollection vector

FIGURE 11:
Feature Space Key Vectors Associatively Processed for a DC-10 at 10° as the Input.

FIGURE 12:
Feature Space Key Vector Output Recollection Vector Data for a Phantom Input at 10°

FIGURE 13:
Feature Space Key Vector Output Recollection Recollection Vector Data for a DC-10 Input at 10°

as iterative AAMs, and AAMs in conjunction with HAMs in a cascade were found useful to reduce noise and improve input image quality. For vision processing, we require: distortion invariance, shift invariance, adaptive updating, processing multiple objects, high clutter performance, etc. We achieve these goals with various new associative memory processors operating on: iconic, feature and symbolic input data. Very successful initial results have been obtained.

ACKNOWLEDGMENTS

The support of this research by a grant from the Air Force Office of Scientific Research is gratefully acknowledged.

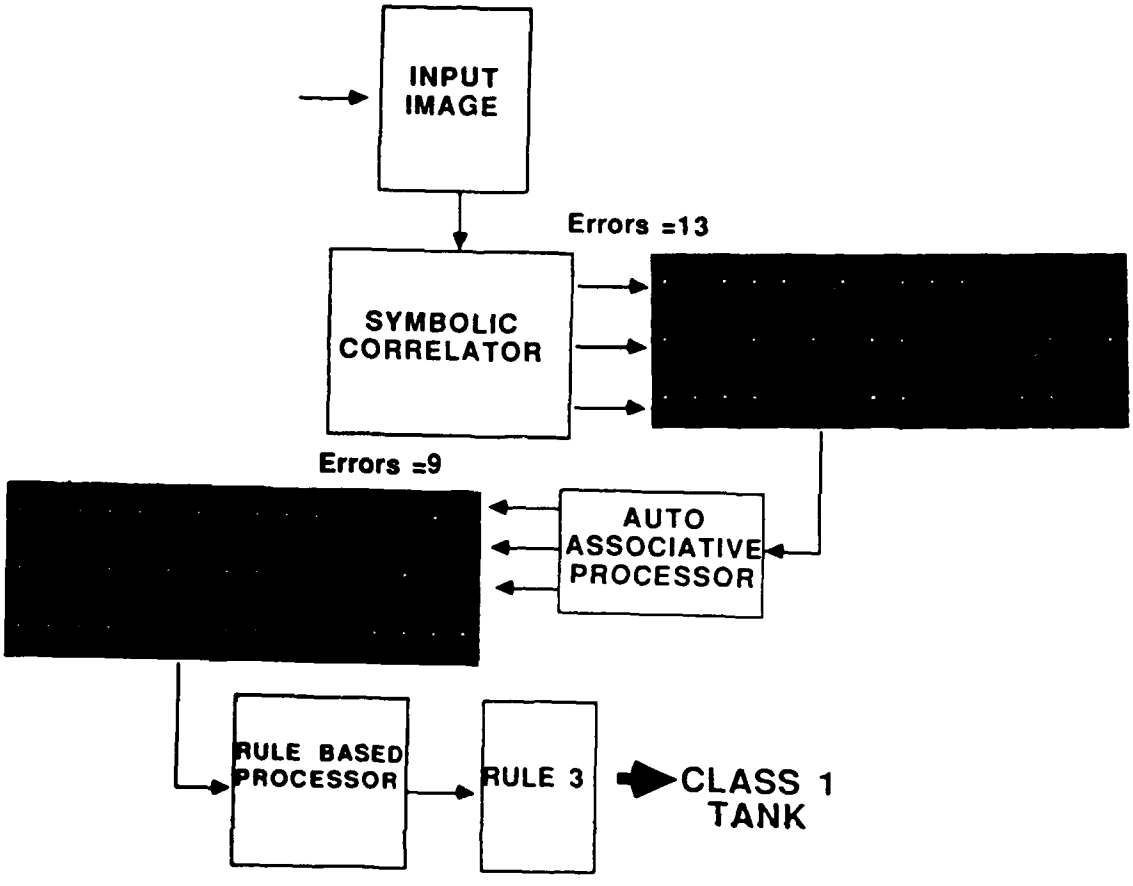


FIGURE 14:
Autoassociative Memory Processing of Symbolic Output Data

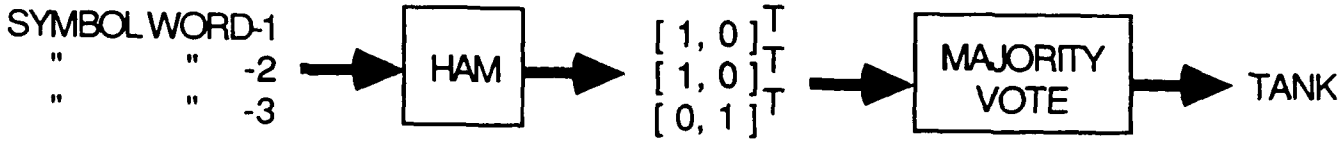


FIGURE 15:
Heteroassociative Memory Processing of Symbolic Object Data for Class Estimation and Updating

REFERENCES

1. T. Kohonen, Self Organization and Associative Memory, Springer-Verlag, New York, 1984.

2. D. Casasent and B. Telfer, "Associative Memory Synthesis, Performance, Storage Capacity and Updating: New Heteroassociative Memory Results", Proc. SPIE, Vol. 848, November 1987.

3. D. Casasent and B. Telfer, "Key and Recollection Vector Effects on Heteroassociative Memory Performance", Applied Optics, submitted.

4. D. Casasent and B. Telfer, "Distortion-Invariant Associative Memories and Processors", Proc. SPIE, Vol. 697, August 1986.

5. D. Casasent, "Scene Analysis Research: Optical Pattern Recognition and Artificial Intelligence", SPIE, *Advanced Institute Series on Hybrid and Optical Computers*, Vol. 634, pp. 439-456, Leesburg, Virginia, March 1986.

6. B. Montgomery and B.V.K. Vijaya Kumar, "An Evaluation of the Use of the Hopfield Neural Network Model as a Nearest-Neighbor Algorithm", Applied Optics, Vol. 25, pp. 3759-3766, 15 November 1986.

7. H. Kasden, "Industrial Applications of Diffraction Pattern Sampling", Optical Engineering, Vol. 18, No. 5, pp. 496-503, September/October 1979.

8. D. Casasent and A. Mahalanobis, "Rule-Based Symbolic Processor for Object Recognition", Applied Optics, Vol. 26, pp. 4795-4802, 15 November 1987.

CHAPTER 10:

"HO-KASHYAP ASSOCIATIVE PROCESSORS"

Proc. SPIE
Vol. 1005
November 1988

Ho-Kashyap Associative Processors

Brian Telfer and David Casasent
Center for Excellence in Optical Data Processing
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

A Ho-Kashyap (H-K) associative processor (AP) is demonstrated to have a larger storage capacity than the pseudoinverse AP and to allow linearly dependent key vectors to be accurately stored. A new Robust H-K AP is shown to perform well over all M/N (where M is the number of keys and N is their dimension), specifically when $M \approx N$, where the standard pseudoinverse and H-K APs perform poorly. Also considered are variable thresholds, an error-correcting algorithm to allow analog synthesis of the H-K AP, and the different reliabilities of the recollection elements.

1 Introduction

The storage capacity and noise performance of APs are of major concern [1] as are key vector requirements [2,3]. It is important to distinguish between general memory and pattern recognition (PR) applications. In a general memory application, APs store arbitrary data and it is fair to assume that the keys and recollections are drawn from random distributions (this allows APs to be tested with Monte Carlo methods). In PR problems, an AP has many key vectors associated with the same recollection vector (a class label), and must generally operate on shifted and distorted input patterns.

In Section 2, we review the pseudoinverse AP and establish our notation. The section also briefly considers the popular class of correlation memories [1], including the Hopfield memory [4] and its variants. Section 3 advances our H-K algorithms. Section 4 gives theoretical and simulation results for the general memory application. A case study of distortion invariant aircraft recognition is presented in Section 5. In Section 6, we offer a summary and conclusion.

2 Pseudoinverse Associative Processor Formulation

Two major APs are the pseudoinverse memory and the broad class of correlation memories, which includes the Hopfield memory. We briefly consider correlation APs, and then review pseudoinverse AP formulation.

Denoting the keys and recollections as vectors $\mathbf{x}_k$ (N-dimensional) and $\mathbf{y}_k$ (K-dimensional) respectively, where $k = 1, ..., M$, the vectors $\mathbf{x}_k$ and $\mathbf{y}_k$ form an associated key/recollection pair (there are M such pairs). We desire a $K \times N$ matrix $\mathbf{M}$ satisfying

$$\mathbf{y}_k = \mathbf{M}\mathbf{x}_k, \quad \text{for } k = 1, ..., M. \tag{1}$$

Defining matrices $\mathbf{X}$ ($N \times M$) and $\mathbf{Y}$ ($K \times M$) with the key and recollection vectors as their columns, (1) can be rewritten as

$$\mathbf{Y} = \mathbf{M}\mathbf{X}. \tag{2}$$

It is useful to distinguish between autoassociative processors (AAPs), in which $\mathbf{Y} = \mathbf{X}$, and heteroassociative processors (HAPs). Autoassociative processors are used for restoring partial or noisy inputs. Our major concern is HAPs since they are useful for decisions and PR.

The correlation memory is given as

$$\mathbf{M} = \mathbf{Y}\mathbf{X}^T, \tag{3}$$

which is an exact solution of (2) only when the keys are orthonormal. The Hopfield memory is a particular correlation AAP that stores binary keys. It is characterized by iterative recall in which the output vector is passed through a threshold and fed back to the input until it converges. As originally proposed [4], the memory matrix has a zero diagonal. Many variants have been proposed, e.g. [5,6]. The Hopfield memory has been shown [4] empirically to have a capacity of $M \approx 0.15N$. Theoretically, an asymptotic ($N \rightarrow \infty$) capacity of $M = N/(4log_2N)$ has been shown [7]. Because of its very low capacity, we do not further consider this or similar APs.

A more general solution of (2) is [1]

$$\mathbf{M} = \mathbf{Y}\mathbf{X}^+, \tag{4}$$

where $\mathbf{X}^+$ is the pseudoinverse of $\mathbf{X}$. This solution minimizes the squared error $||\mathbf{Y}-\mathbf{M}\mathbf{X}||^2$. When the key vectors are linearly independent, the squared error is zero and (4) solves (1). Thus, when a key vector is input, the exact recollection is output. We find orthogonal and linearly independent key vector requirements to be unrealistic. When the key vectors are linearly dependent, the solution in (4) is approximate. Because perfect recall is desired, the pseudoinverse AP in (4) is often restricted to the case $M \leq N$ (since when $M > N$, the keys are guaranteed to be linearly dependent). However, the approximate solutions are also useful and allow a larger storage capacity. They have generally not been considered. We will refer to (4) as an exact pseudoinverse AP (when $M \leq N$ and the keys are linearly independent) and as an approximate pseudoinverse AP (when $M \geq N$).

The maximum storage capacity of the exact pseudoinverse AP is limited by definition to $M = N$ for both general memory (random keys and recollections) and PR (many keys associated with the same recollection) applications. The maximum storage capacity of the approximate pseudoinverse AP is difficult to quantify for the general memory application, but only exceeds $M = N$ by a small margin. The maximum storage capacity of the H-K AP is $M = 2N$ for general memory applications, as will be discussed in Section 4. The storage capacity for PR is quite different than for a general memory. A PR HAP can have $M \gg N$ and the issue is that the key vectors should represent the distortions of the different classes. In PR APs, recall accuracy for a specific application is important rather than storage capacity.

A variety of algorithms exist for computing the pseudoinverse [1,8]. In this paper we use the singular value decomposition (SVD) approach [9] because it can be used for either linearly independent or dependent keys and because it allows us to improve the AM's noise performance by a method that will be explained below. If $M \leq N$, the decomposition of $\mathbf{X}$ is given as

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{5}$$

where $\mathbf{U}$ is an $N \times M$ matrix with orthonormal columns, $\mathbf{V}$ is an $M \times M$ orthogonal matrix, and $\mathbf{\Sigma}$ is an $M \times M$ diagonal matrix that contains the singular values $\mu_i, i = 1, ..., M$. If $M \geq N$, the decomposition of Eq. 5 still holds, but $\mathbf{U}$ is now an $N \times N$ orthogonal matrix, $\mathbf{V}$ is an $M \times N$ matrix with orthonormal columns, and $\mathbf{\Sigma}$ is an $N \times N$ diagonal matrix that contains the singular values $\mu_i, i = 1, ..., N$. The pseudoinverse of $\mathbf{X}$ is expressed as

$$\mathbf{X}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T, \tag{6}$$

where $\mathbf{\Sigma}^+ = diag(\mu_i^+)$ and

$$\mu_i^+ = \begin{cases} 1/\mu_i & \text{for } \mu_i > 0 \\ 0 & \text{for } \mu_i = 0. \end{cases}$$

The conventional pseudoinverse HAP recalls noisy key vectors poorly when $M \approx N$ (with much better performance occurring when $M < N$ and $M > N$) [10]. A recent paper [11] explains this phenomemon and shows that to optimize recall accuracy for a particular noise variance σ^2, all singular values μ_i such that

$$\mu_i < \sqrt{M}\,\sigma \tag{7}$$

should be set to zero. Then the memory matrix is computed by [11]

$$\mathbf{M} = \mathbf{Y}\tilde{\mathbf{X}}^+, \tag{8}$$

where $\tilde{\mathbf{X}}$ is the modified key matrix with small singular values set to zero. For realistic σ values, this method causes only a small decrease in the recall accuracy for exact key vector inputs, and significantly improves recall accuracy when noise is present (especially for $M \approx N$). The method does not alter the pseudoinverse HAP's performance for $M \ll N$ and $M \gg N$. We use this approach in our simulations described in Sec. 4. We note that it is well known [9] that very small singular values (e.g. 10^{-4}) should always be set to zero to avoid numerical instability. The method explained above differs from this, in that the threshold for zeroing the singular values is given as a function of M and σ (as opposed to selecting it arbitrarily) and that the singular values zeroed out can exceed 10^{-4} by orders of magnitude (e.g., if $M = 50$ and $\sigma = 0.1$, the threshold for μ is 0.71).

3 Ho-Kashyap (H-K) Associative Processors (APs)

The H-K AP has a larger storage capacity than the pseudoinverse AP because it requires that the key vectors be only linearly separable for perfect recall, rather than linearly independent, as is the pseudoinverse AP requires. Since linear separability is a looser restriction than linear independence, the H-K AP in many cases can perfectly store linearly dependent keys. Before presenting the H-K algorithm, we first describe how linear separability applies to APs.

3.1 Linear Separability and the H-K Algorithm

We consider the case where $\mathbf{Y}$ is bipolar binary. The columns of $\mathbf{Y}$ are the recollection vectors for the different key vectors and row i of $\mathbf{Y}$ gives the desired values of the i-th output element for the different key vectors. With binary recollection elements, the output vector is passed through a threshold. If an input is a distorted key, or if the keys are linearly dependent, then the pre-threshold analog output of the pseudoinverse AP is likely to differ from the desired result, but it is still possible for the post-threshold binary output to be correct. Each row of the AP matrix, with its threshold value, forms a linear discriminant function (LDF) that separates the N-dimensional input space with a hyperplane into two classes, those key vectors for which element i of the recollection vector is -1 and those for which it is +1. The locations of the ± 1 elements in row i of $\mathbf{Y}$ denote these two classes for that row. If these two classes can be separated with a hyperplane, then they are linearly separable and there exists an LDF that will give perfect recall for that output recollection element. The pseudoinverse AP minimizes the squared error, but is not guaranteed to give perfect recall even if the K class groupings (one for each output recollection vector element) are all linearly separable [12]. The Ho-Kashyap algorithm iteratively computes an LDF (i.e. one row of $\mathbf{M}$) that will correctly classify two classes if they are linearly separable [12]. If they are not linearly separable, the algorithm will still converge to a minimum squared error solution, and will indicate that the classes are not linearly separable.

3.2 H-K APs

The H-K algorithm is noted in Table 1 as a new matrix version for AP synthesis. We begin with an estimate of $\mathbf{M}$ from the pseudoinverse (step 1). We modify $\mathbf{Y}$ (step 4) and $\mathbf{M}$ (step 1) in successive iterations. In Table 1, $\mathbf{S}$ contains the signs of the $\mathbf{Y}$ elements, $\otimes$ denotes Hadamard (pointwise) multiplication, and the subscript n is the iteration index. If the pseudoinverse is exact (i.e., the keys are linearly independent) then no modifications will be made. The H-K algorithm improves the pseudoinverse memory when the keys are linearly dependent. In step 2, we calculate the error matrix $\mathbf{E}$, which gives the errors between the actual and desired outputs. In step 3, we form a modified error matrix $\mathbf{E}'$. This matrix equals $\mathbf{E}$ except that all $\mathbf{E}$ elements that differ in sign from the corresponding elements in $\mathbf{Y}$ are set to 0. This ensures that none of the $\mathbf{Y}$ elements change sign (we assume initial bipolar binary $\mathbf{Y}$ values) when $\mathbf{E}'$ is added to $\mathbf{Y}$ in step 4 to produce an updated $\mathbf{Y}$. Step 5 then returns the algorithm to step 1 where $\mathbf{M}$ is updated.

Once $\mathbf{E}'_n = \mathbf{0}$, the algorithm has converged (convergence is guaranteed). If a row of $\mathbf{E}_n$ equals 0 then that

Step			Operation
1	$\mathbf{M}_n$	=	$\mathbf{Y}_n\mathbf{X}^+$
2	$\mathbf{E}_n$	=	$\mathbf{Y}_n - \mathbf{M}_n\mathbf{X}$
3	$\mathbf{E}'_n$	=	$\mathbf{S} \otimes \frac{1}{2}[(\mathbf{S} \otimes \mathbf{E}_n) + \mid \mathbf{S} \otimes \mathbf{E}_n \mid]$
4	$\mathbf{Y}_{n+1}$	=	$\mathbf{Y}_n + 2\rho\mathbf{E}'_n, \quad 0 < \rho < 1$
5	If $\mathbf{E}'_n \neq \mathbf{0}$ go to 1.		

Table 1: Ho-Kashyap AP Algorithm

row's dichotomy (grouping into two classes) is linearly separable; otherwise it is not. In actual application, the algorithm can also be stopped if M correctly recalls all of the key vectors.

Other AP work [13-15] used the H-K algorithm for computing APs, but limited the number of key vectors to be substantially less than their dimensionality. (An overdetermined problem was created by adding key/recollection vector constraints to map unit vectors (with a single 1) to all-zero vectors, in addition to storing the key vectors.) A major emphasis of this paper is that the advantage of the H-K AP over the pseudoinverse AP occurs when the number of key vectors exceeds their dimensionality and that the H-K AP can handle linearly dependent key vectors (i.e. $M > N$). Our memory matrices are overdetermined by the large number of keys that we store, and we do not need additional constraints (as used in other work) to create an overdetermined solution.

Other H-K AP work also used output thresholding and feedback in recall mode. We do not consider this for high capacity APs ($M > N$) for the following reasons. To use feedback in a HAP recall requires a Bidirectional Associative Memory (BAM) [16]. To use the H-K algorithm to synthesize a BAM [15] requires that both forward (key $\rightarrow$ recollection) and reverse (recollection $\rightarrow$ key) mappings be computed, which is a significant computational increase over the unidirectional processor synthesis. For perfect recall, if the reverse mapping is not linearly separable, it can introduce errors into the BAM recall. Also, the capacity of the BAM is limited by the minimum of N and K [16]. When K is small (which is to be expected when the recollections are used for decisions or class labelling), it will cause the memory's capacity to be less than that for a unidirectional processor.

It is well known that the pseudoinverse AAP degenerates to the identity matrix when the rows of $\mathbf{X}$ are linearly independent [1], which is likely to occur when $M > N$. Although this is clearly not a useful processor, it does correctly recall exact key inputs, and the H-K algorithm cannot improve on it. This is another reason why this paper considers only HAPs.

3.3 Most Reliable Recollection Vector Elements

Since the final $\mathbf{E}$ indicates which output elements give perfect recall for the key vector inputs, the H-K algorithm automatically provides information about which output elements are the most reliable. If the data are linearly separable, $\mathbf{E}$ and $\mathbf{E}'$ will be all zero (and the new $\mathbf{Y}$ and $\mathbf{M}$ achieve this linear separation). If $\mathbf{E}'$ is all zero and $\mathbf{E}$ is not, at least one row of $\mathbf{Y}$ is not linearly separable (but the resultant $\mathbf{M}$ is better than the approximate pseudoinverse solution, in that it has a lower squared error). The rows of $\mathbf{E}$ and $\mathbf{E}'$ with all zero elements denote the output elements that are reliable. This allows us to consider the reliable outputs first and then the other elements with a reduced confidence algorithm. We do not consider this topic further in this paper, but will address it in future work.

3.4 Robust H-K AP

Our basic H-K AP (Table 1) uses the exact or approximate pseudoinverse AP as an initial solution and then refines it. When the pseudoinverse is used, as in the basic Ho-Kashyap algorithm, then the resulting memory will suffer

from the same recall deficiencies as the pseudoinverse memory when $M \approx N$ (this will be shown in Sec. 4). We therefore propose that $\tilde{\mathbf{X}}^+$ be used instead of $\mathbf{X}^+$ in Table 1. We call this the Robust H-K AP. This combination of the H-K algorithm and setting small eigenvalues to zero is quite new.

3.5 Error Correcting H-K AP Algorithm

We now mention the use of an error correcting H-K algorithm [17] that can be used to produce a new error correcting H-K AP algorithm, which does not require an initial $\mathbf{X}^+$. The algorithm can be applied in cases where the *rows* of $\mathbf{X}$ are linearly independent, which is likely to occur when $M > N$. Thus, it is applicable for linearly dependent key vectors (the *columns* of $\mathbf{X}$) and the high capacity memories that we consider. Because of its error correcting nature and the fact that SVD is not used, we expect it to tolerate lower accuracy than the first H-K algorithm (Table 1). Hence it appears attractive for optical implementation. The algorithm updates $\mathbf{Y}$ and $\mathbf{M}$ using

$$\begin{aligned} \mathbf{Y}_{n+1} &= \mathbf{Y}_n + 2\rho\mathbf{E}'_n \\ \mathbf{M}_{n+1} &= \mathbf{M}_n + 2\rho\mathbf{E}'_n\mathbf{X}^T\mathbf{R}, \end{aligned} \tag{9}$$

where $\mathbf{R}$ can be any positive definite matrix. The simplest choice is $\mathbf{R} = \mathbf{I}$. If $\mathbf{R} = (\mathbf{X}\mathbf{X}^T)^{-1}$, then this becomes the first H-K algorithm (since $\mathbf{X}^+ = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$ when the rows of $\mathbf{X}$ are linearly independent).

3.6 Augmented APs

In our PR case study (Section 5), we augment each of the key vectors with a "1." This increases the key vector hyperspace from N to $N+1$ dimensions and does not require the separating hyperplanes to pass through the origin, as would occur otherwise. This technique is well known [12,18] and in APs is equivalent to varying the thresholds [13].

4 General Memories

We first review theoretical work and then report our simulation results.

4.1 Theoretical Results

Classic theoretical results allow us to estimate the storage capacity of the H-K AP. The results assume that the M N-dimensional key vectors are in general position. For a group of points to be in general position, no subset of N vectors can be linearly dependent. Thus, restricting points to be in general condition is a looser condition than linear independence. There are 2^M possible dichotomies of these vectors. The fraction of these that are linearly separable is [12,18]

$$f(M,N) = \begin{cases} 1 & M \le N \\ 2^{1-M}\sum_{i=0}^{N-1}\binom{M-1}{i} & M > N. \end{cases} \tag{10}$$

When the keys cannot be assumed to be in general position, (10) is an upper bound. We extended (10) to an associative memory formulation (with K-element recollection vectors), by finding the fraction of groups of K dichotomies that are all linearly separable. This gives the fraction of all possible $\mathbf{Y}$ matrices that can be correctly recalled in an H-K AP. Our fraction is (10) raised to the K-th power:

$$g(M,N,K) = \begin{cases} 1 & M \le N \\ 2^{K-MK}\left[\sum_{i=0}^{N-1}\binom{M-1}{i}\right]^K & M > N. \end{cases} \tag{11}$$

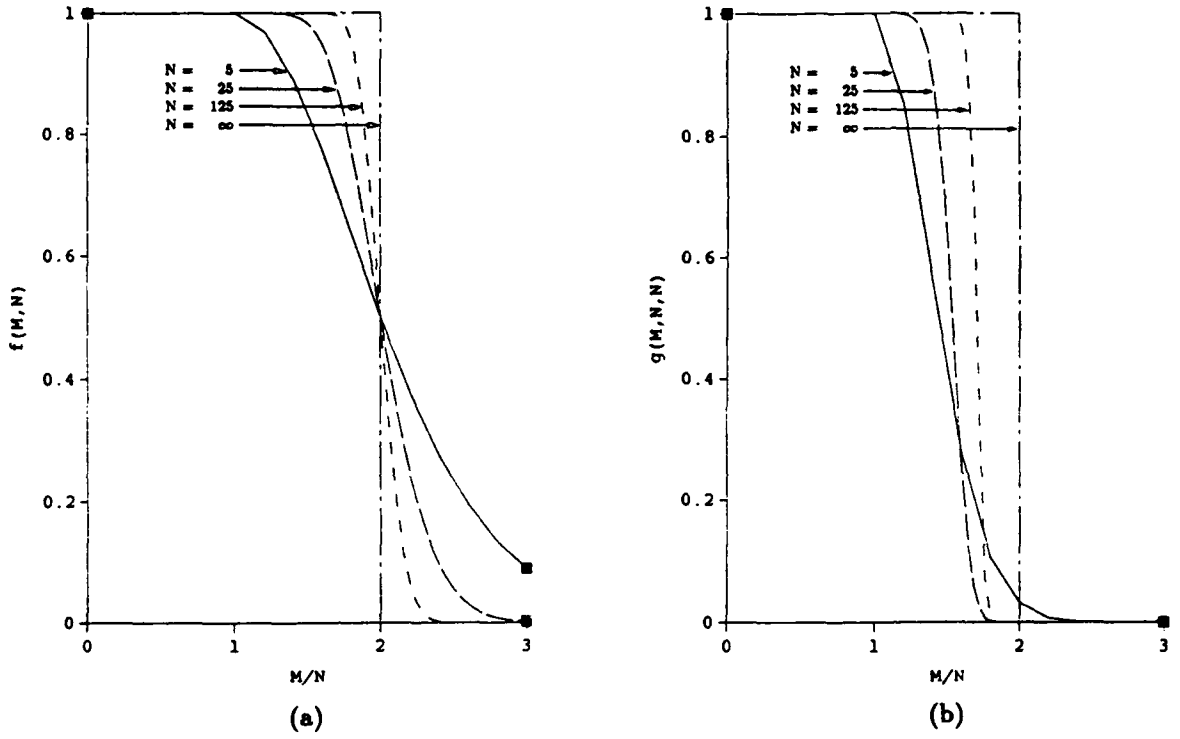


Figure 1: Fraction of (a) all dichotomies of M N-dimensional points that are linearly separable, and (b) all groups of K dichotomies of M N-dimensional points that are linearly separable

The asymptotic limit (for $K = N$) as $N \rightarrow \infty$ is (modified from [19])

$$g(M, N, N) = \begin{cases} 1 & M/N < 2 \\ 0 & M/N \geq 2. \end{cases} \tag{12}$$

As seen, the maximum storage for an H-K AP in a general memory application (random keys and recollections) is

$$M = 2N. \tag{13}$$

This asymptotic limit is not achievable with finite length (N) key vectors. As always, better storage (larger M) is achieved with larger sized key vectors (larger N).

Fig. 1a shows (10) vs. M/N. It shows the probability that one row of $\mathbf{Y}$ is linearly separable. Fig. 1b shows (11) vs. M/N for $K = N$. It shows the probability that all K rows of $\mathbf{Y}$ are linearly separable. As seen, the maximum storage capacity of $M = 2N$ cannot be achieved except with infinite N, but (11) allows us to estimate the storage capacity for finite N.

4.2 H-K and Pseudoinverse General Memory Simulations

We now test random H-K APs for agreement with the above theory and for comparison to pseudoinverse APs. We use $N = 50$ element key vectors, $K = N = 50$ element recollection vectors and vary M/N. When $M/N > 1$, the

key vectors are automatically linearly dependent. All key vectors were randomly chosen and uniformly distributed over -1 to +1. Each bipolar binary recollection vector element was chosen randomly to be -1 or +1. For each M/N value, one X and one Y matrix were generated. The H-K synthesis algorithm used $\rho = 0.5$ and was limited to a maximum of 1000 iterations. The algorithm was also stopped when the memory perfectly recalled the exact key inputs or when $\mathbf{E}' = \mathbf{0}$. To test each AP, we used M key vectors with four levels of additive zero-mean Gaussian noise: $\sigma = 0.0, 0.1, 0.2, 0.4$. The recall accuracy (percentage of correct output elements) was computed for each noise level. Fig. 2 shows the results for the standard pseudoinverse AP (Fig. 2a) and the results for our H-K AP (Fig. 2b), with M/N varied from 0.0 to 2.0 in 0.2 increments. These results show improved performance and storage capacity (with and without noise) for the H-K vs. the pseudoinverse AP. The pseudoinverse AP performs perfectly for exact key inputs only up to $M = N$, and degrades for $M > N$. The H-K AP achieves perfect recall for $M \leq 1.4N$ for inputs with no noise. This is a 40% improvement of the H-K over the pseudoinverse AP. Thus, the H-K AP performs significantly better than the pseudoinverse AP. Its improvement over the Hopfield AP ($M = 0.15N$) is a factor of 10 or 1000% better performance. We note that in all cases, noise performance degrades when $M \approx N$ (as is typical of pseudoinverse APs).

We now consider the use of our robust H-K AP to improve noise performance when $M \approx N$ in Fig. 2. The performance of the robust pseudoinverse AP and the robust H-K AP (both of which have small eigenvalues removed) are shown in Figs. 3a and 3b respectively, with M/N varied from 0.6 to 1.6 with 0.04 increments. As seen, both APs avoid the severe drop in performance (when $M \approx N$) in Fig. 2 for noisy inputs. For the modified APs with exact key inputs, we obtained:

$P_c = 100\%$ for $M \leq 0.76N$ for the robust pseudoinverse AP,
$P_c = 100\%$ for $M \leq 1.48N$, for the robust H-K AP,
90% storage improvement with robust H-K AP over robust pseudoinverse AP.

From Table 2, we see that H-K improves the modified pseudoinverse even when $M \geq 0.8N$ (for $M < 0.8N$, the pseudoinverse is exact because no eigenvalues were set to zero). The number of iterations (Table 2) jumped for $M = 1.48N$, since here the data became more difficult to linearly separate. For $M \geq 1.6N$, the H-K algorithm did not converge within 1000 iterations, which indicates that at least one row of the memory matrix was probably not linearly separable. Table 3 notes the rank of the modified $\mathbf{X}^+$ (with small eigenvalues set to zero). As seen, five eigenvalues are omitted at $M = N$ and the matrix is full rank (i.e. no eigenvalues were set to zero) for $M \geq 1.44N$.

We note good agreement between theory and tests. In our tests, the transition from a completely linearly separable memory to an at least partially linearly nonseparable memory occurred from $M = 1.48N$ (completely separable) to $M = 1.52N$ (partially nonseparable). (These numbers refer to the Robust H-K AP tests, but for $M \geq 1.44N$ the Robust H-K AP was equivalent to the H-K AP, since $\tilde{\mathbf{X}}^+$ was a full rank matrix.) According to (11), for $N = K = 50$, this same transition theoretically takes place roughly between $M = 1.5N$ and $M = 1.7N$, with the probability of a completely linear separable memory for these two capacities being 0.92 and 0.07 respectively (these values are not shown in Fig. 1b but were computed from (11)). The experimental transition occurred at a slightly lower capacity than the theoretical transition. This is to be expected since the theoretical transition is an upper bound due to its general position assumption. The two transitions still agree reasonably well. Thus, (11) allows us to estimate the H-K AP's capacity for finite N.

M/N	0.60	0.64	0.68	0.72	0.76	0.80	0.84	0.88	0.92	0.96	1.00	1.04	1.08
iterations	0	0	0	0	0	1	2	2	1	2	4	3	5
M/N	1.12	1.16	1.20	1.24	1.28	1.32	1.36	1.40	1.44	1.48	1.52	1.56	1.60
iterations	5	3	12	15	64	60	45	80	46	566	1000	1000	1000

Table 2: Number of iterations required by modified Ho-Kashyap algorithm for different M/N

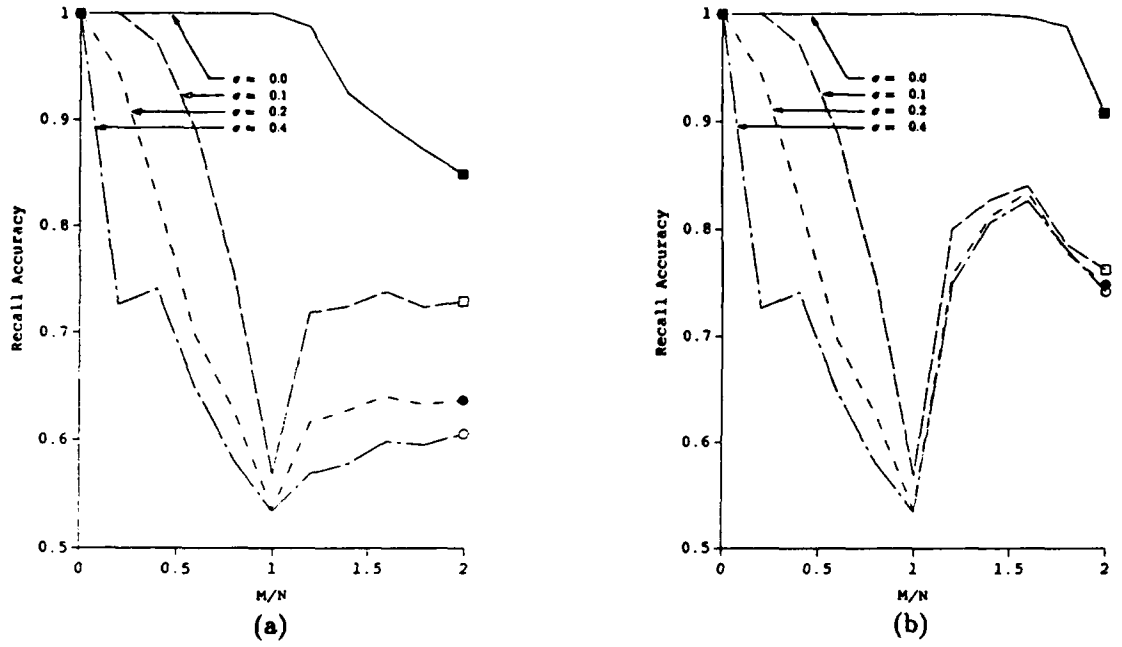


Figure 2: Recall accuracy vs. M/N for exact and noisy key vector inputs using a) conventional pseudoinverse associative memory and b) conventional Ho-Kashyap associative memory

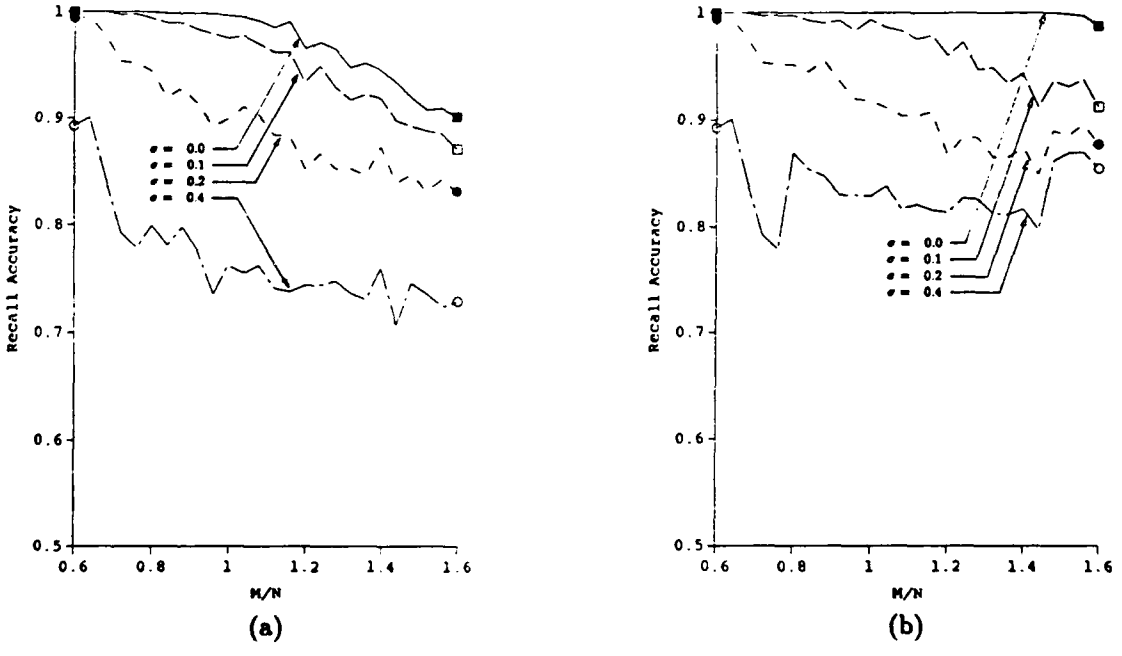


Figure 3: Recall accuracy vs. M/N for exact and noisy key vector inputs using a) robust pseudoinverse associative memory and b) robust Ho-Kashyap associative memory

M/N	0.60	0.64	0.68	0.72	0.76	0.80	0.84	0.88	0.92	0.96	1.00	1.04	1.08
rank	30	32	34	36	38	39	40	41	42	44	45	45	46
M/N	1.12	1.16	1.20	1.24	1.28	1.32	1.36	1.40	1.44	1.48	1.52	1.56	1.60
rank	46	48	48	49	49	49	50	49	50	50	50	50	50

Table 3: Rank of modified key matrix for different M/N

5 Aircraft Recognition Case Study

This section presents a comparison of the pseudoinverse and Ho-Kashyap AP approaches in a two-class aircraft recognition problem. We consider two classes (Phantom and DC-10) of 128 x 128 pixel aircraft imagery. As our key vector representation space, we use 32 wedge samples of the Fourier transform (in half of the transform plane). The wedge feature space provides scale invariance (when the wedge samples are normalized) and shift invariance, and is easily generated optically [20]. In-plane image rotations cause the wedge samples to circularly shift. We postulate that the aircraft are moving, and that tracking information provides the location of the aircraft's nose. This information allows the wedge samples from an unidentified aircraft to be circularly shifted so that they align properly with the training vectors. Thus, for moving aircraft this feature space is rotation (in-plane), scale and shift invariant. Thus, we do not test these invariances. We augment each of the key vectors with a "1," as described in Section 3.6, to improve recall accuracy. The recollection vectors are of dimension $K = 1$ with values of +1 for the Phantom and -1 for the DC-10.

Two training sets were tested. The first consists of 882 key vectors produced from the two aircraft rotated in pitch and roll between $\pm 50^\circ$ at 5° increments. The second consists of 1250 key vectors produced from the two aircraft rotated in pitch and roll between $\pm 60^\circ$ at 5° increments. Thus, we consider APs with

$N = 33$ dimensional key vectors,
$M = 882$ and 1250 key/recollection pairs,
$K = 1$ dimensional recollection vectors.

Both cases represent linearly dependent key vectors, with $M = 25N$ and $M = 35N$ respectively. An H-K AP was produced using the same stopping criteria as in Section 4 (we did not test the robust H-K algorithm since the test vector distortions are not easily quantified into an equivalent σ for noise). The first case ($M = 882$) represents a linearly separable problem, since H-K gives 100% correct classification (after 32 iterations). As shown in Table 4, the H-K AP yields perfect performance on the training set, whereas the pseudoinverse AP does not. The test data in Table 4 are 800 aircraft (not present in the training set) with pitch and roll varied between $\pm 57.5^\circ$ at 5° increments (i.e. at least 2.5° different in pitch and roll from the training data). The H-K AP also gives perfect performance for these inputs. The second case ($M = 1250$) represents a linearly nonseparable problem, as is shown in Table 5. The H-K algorithm was stopped at 1000 iterations. The test data in Table 5 are 1152 aircraft with pitch and roll varied between $\pm 57.5^\circ$ at 5° increments. We see that H-K gives excellent performance in both cases.

% Misclassified		
Training Set:	Pseudoinverse	0.68
	H-K AP	0.00
Test Set:	Pseudoinverse	0.25
	H-K AP	0.00

Table 4: Misclassification results for pseudoinverse and Ho-Kashyap memories with $\pm 50^\circ$ training set

% Misclassified		
Training Set:	H-K AP	1.04
Test Set:	H-K AP	0.52

Table 5: Misclassification results for pseudoinverse and Ho-Kashyap memories with $\pm 60°$ training set

6 Summary and Conclusion

We have shown that the Ho-Kashyap associative processor has a larger storage capacity than the pseudoinverse processor and that it can handle linearly dependent key vectors. We have detailed a new Robust Ho-Kashyap processor to improve the noise performance of the H-K AP. These new processors allow operation on linearly dependent key vectors, achieve much better storage ($M \approx 2N$ for general memory applications), and improved noise performance when $M \approx N$. We showed: 100% recall accuracy for our Ho-Kashyap general memories for $M \leq 1.4N$ and 99.7% accuracy with $M = 1.6N$; over 1000% better performance than the Hopfield memory; 40% better performance than the pseudoinverse memory; and 90% improved noise performance when $M \approx N$. Our pattern recognition case study showed 3-D distortion invariance and excellent (> 99%) recall accuracy for large $M = 25N$ and $M = 35N$ cases.

Acknowledgements

The support of this research by the Air Force Office of Scientific Research (Grant AFOSR-84-0293) and the partial support by the internal research and development funds of General Dynamics (Agreement No. 752647) is gratefully acknowledged.

References

[1] Teuvo Kohonen. *Self-Organization and Associative Memory, 2nd edition.* Springer-Verlag, Berlin, 2nd edition, 1988.

[2] David Casasent and Brian Telfer. Associative memory synthesis, performance, storage capacity and updating: New heteroassociative memory results. In David P. Casasent and Ernest L. Hall, editors, *Intelligent Robots and Computer Vision*, pages 313–333. Proc. SPIE, vol. 848, 1987.

[3] David Casasent and Brian Telfer. Key and recollection vector effects on heteroassociative memory performance. *accepted for publication by Applied Optics.*

[4] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA*, 79:2554–58, April 1982.

[5] B. Macukow and H.H. Arsenault. Optical associative memory model based on neural networks having variable interconnection weights. *Applied Optics*, 26(5):924–28, 1 Mar. 1987.

[6] G.R. Gindi, A.F. Gmitro, and K. Parthasarathy. Hopfield model associative memory with nonzero-diagonal terms in memory matrix. *Applied Optics*, 15 June 1988.

[7] R.J. McEliece et. al. The capacity of the hopfield associative memory. *IEEE Trans. Info. Theory*, IT-33(4):461–82, July 1987.

[8] B. Telfer and D. Casasent. Updating optical pseudoinverse associative memories. *submitted to Applied Optics.*

[9] Gilbert Strang. *Linear Algebra and Its Applications.* Harcourt Brace Jovanovich, Orlando, 1980.

[10] G.S. Stiles and Dong-Lih Denq. On the effect of noise on the Moore-Penrose generalized inverse associative memory. *IEEE Trans. Pat. Anal. and Mach. Int.*, PAMI-7(3):358–60, May 1985.

[11] Kenji Murakami and Tsunehiro Aibara. An improvement on the Moore-Penrose generalized inverse associative memory. *IEEE Trans. Sys. Man and Cybern.*, SMC-17(4):699–707, July/Aug. 1987.

[12] R.O. Duda and P.E. Hart. *Pattern Classifiation and Scene Analysis.* John Wiley and Sons, New York, 1973.

[13] M.H. Hassoun and A.M. Youssef. New recording algorithm for hopfield associative memories. In *Proc. SPIE, vol. 882*, 1988.

[14] M.H. Hassoun and D.W. Clark. An adaptive attentive learning algorithm for single-layer neural networks. In *Proc. IEEE International Conf. on Neural Networks*, pages 431–40 (Vol. I), San Diego, July 1988.

[15] M.H. Hassoun. A high-performance associative neural memory (ANM) for pattern recognition. In *Piece Recognition and Image Processing.* Proc. SPIE, Vol. 956.

[16] B. Kosko. Adaptive bidirectional associative memories. *Applied Optics*, 26(23):4947–60, 1 Dec. 1987.

[17] Y.-C. Ho and R.L. Kashyap. A class of iterative procedures for linear inequalities. *J. SIAM Control*, 4(1):112–15, 1966.

[18] T.M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Elec. Comp.*, EC-14:326–34, June 1965.

[19] R.O. Winder. Bounds on threshold gate realizability. *IEEE Trans. Elec. Comp.*, EC-12(5):561–64, Oct. 1963.

[20] H. Kasden. Industrial applications of diffraction pattern sampling. *Opt. Eng.*, 18(5):496–503, Sept./Oct. 1979.

CHAPTER 11:

PUBLICATIONS, PRESENTATIONS AND THESES PRODUCED

11.1 AFOSR PUBLICATIONS (August 1984 - April 1989)

1. "Hierarchical Pattern Recognition Using Parallel Feature Extraction", Proc. ASME, *Computers in Engineering 1984*, Vol. 1, pp. 1-6, August 1984 (CASASENT, Cheatham).

2. "SDF Control of Correlation Plane Structure for 3-D Object Representation and Recognition", Proc. SPIE, 507, pp. 9-18, August 1984 (Chang, CASASENT, Fetterly).

3. "Feature Extractors for Distortion-Invariant Robot Vision", Optical Engineering, 23, pp. 492-498, September/October 1984 (CASASENT, Sharma).

4. "Optimal Linear Discriminant Functions", Proc. SPIE, 519, pp. 50-55, October 1984 (Sharma, CASASENT).

5. "Optical Processing Research Making Significant Advancements", Laser Focus, pp. 150, October 1984 (CASASENT).

6. "Optimality Considerations in Modified Matched Spatial Filters", Proc. SPIE, 519, pp. 85-93, October 1984 (Kumar, Pochapsky, CASASENT).

7. "Projection Synthetic Discriminant Function Performance", Optical Engineering, Vol. 23, pp. 716-720, November 1984 (CASASENT, Rozzi, Fetterly).

8. "Chord Distributions in Pattern Recognition: Distortion-Invariance and Parameter Estimation", Proc. SPIE, 521, pp. 2-6, November 1984 (Chang, CASASENT).

9. "Coherent Optical Pattern Recognition: A Review", Optical Engineering, 24, *Special Issue on Optical Computing*, pp. 26-32, January 1985 (CASASENT).

10. "A Computer Generated Hologram for Diffraction-Pattern Sampling", Proc. SPIE, 523, January 1985 (CASASENT, Song).

11. "Hybrid Optical/Digital Image Pattern Recognition: A Review", Proc. SPIE, Vol. 528, pp. 64-82, January 1985 (CASASENT).

12. "Computer Generated Holograms in Pattern Recognition: A Review", Proc. SPIE, Vol. 532, pp. 106-118, January 1985 (CASASENT).

13. "Parallel Coherent Optical Processor Architectures and Algorithms for ATR", Proc. of the Workshop on Algorithm-Guided Parallel Architectures for Automatic Target Recognition, Leesburg, Virginia, July 1984, Published February 1985, pp. 33-49 (CASASENT).

14. "Hierarchical Feature-Based Object Identification", OSA Topical Meeting on Machine Vision, pp. ThD4-1 - ThD4-4, March 1985 (CASASENT, Cheatham).

15. "Correlation Filters for Distortion-Invariance and Discrimination", OSA Topical Meeting on Machine Vision, pp. FB5-1 - FB5-3, March 1985 (CASASENT, Mahalanobis).

16. "Correlation Synthetic Discriminant Functions for Object Recognition and Classification in High Clutter", Proc. SPIE, 575, August 1985 (CASASENT, Rozzi, Fetterly).

17. "Optical Processing Techniques for Advanced Intelligent Robots and Computer Vision", Proc. SPIE, Vol. 579, pp. 208-214, September 1985 (CASASENT).

18. "A High-Dimensionality Pattern Recognition Feature Space", Proc. SPIE, Vol. 579, pp. 245-257, September 1985 (CASASENT, Okuyama).

19. "Computer Generated Holograms in Pattern Recognition: A Review", Optical Engineering, 24, pp. 724-730, September/October 1985 (CASASENT).

20. "Performance of Direct and Iterative Algorithms on an Optical Systolic Processor", Applied Optics, 24, pp. 3883-3892, 15 November 1985 (Ghosh, CASASENT, Neuman).

21. "Frequency-Domain Synthesis of Modified MSFs", Optics Letters, Vol. 10, pp. 517-519, November 1985 (Rozzi, CASASENT).

22. "Optical Computer Architectures for Pattern Analysis", *IEEE Computer Society Workshop* on Computer Architectures for Pattern Analysis and Image Database Management, Miami Beach, Florida, November 1985, pp. 49-52, IEEE Catalog No. 85CH2229-3, ISBN 0-8186-0662-2 (CASASENT).

23. "Modified MSF Synthesis by Fisher and Mean-Square Error Techniques", Applied Optics, Vol. 25, pp. 184-187, 15 January 1986 (CASASENT, Rozzi).

24. "A Feature Space Rule-Based Optical Relational Graph Processor", Proc. SPIE, Vol. 625, pp. 234-243, January 1986 (CASASENT, Lee).

25. "Optical AI Symbolic Correlators: Architecture and Filter Considerations", Proc. SPIE, Vol. 625, pp. 220-225, January 1986 (CASASENT).

26. "Diffraction Pattern Sampling Using a Computer-Generated Hologram", Applied Optics, Vol. 25, pp. 983-989, 15 March 1986 (CASASENT, Xia, Song, Lee).

27. "Scene Analysis Research: Optical Pattern Recognition and Artificial Intelligence", SPIE, *Advanced Institute Series on Hybrid and Optical Computers*, Vol. 634, pp. 439-456, Leesburg, Virginia, March 1986 (CASASENT).

28. "Model-Based System for On-Line Affine Image Transformations", Proc. SPIE, Vol. 638, pp. 66-75, March-April 1986 (CASASENT, Liebowitz).

29. "Optical Computing at Carnegie-Mellon University", Optics News, *Special Issue on Optical Computing*, Vol. 12, pp. 11-13, April 1986 (CASASENT).

30. "Phase Correction of Light Modulators", Optics Letters, Vol. 11, pp. 398-400, June 1986 (CASASENT, Xia).

31. "Optical Artificial Intelligence Processors", *IOCC-1986 International Optical Computing Conference* (Israel), Proc. SPIE, Vol. 700, July 1986, pp. 246-250, 1986 (CASASENT).

32. "Correlation Synthetic Discriminant Functions", Applied Optics, Vol. 25, pp. 2343-2350, 15 July 1986 (CASASENT, Chang).

33. "Distortion-Invariant Associative Memories and Processors", Proc. SPIE, Vol. 697, pp. 60-77, August 1986 (CASASENT, Telfer).

34. "Optical Relational-Graph Rule-Based Processor for Structural-Attribute Knowledge Bases", Applied Optics, Vol. 15, , pp. 3065-3070, 15 September 1986 (CASASENT, Lee).

35. "Large Class Iconic Pattern Recognition: An OCR Case Study", Proc. SPIE, Vol. 726, pp. 2-27, October 1986 (Mahalanobis, CASASENT).

36. "Hierarchical Processor and Matched Filters for Range Image Processing", Proc. SPIE, Vol. 727, pp. 180-203, October 1986 (Liebowitz, CASASENT).

37. "Computer-Generated and Phase-Only Synthetic Discriminant Function Filters", Applied Optics, Vol. 25, pp. 3767-3772, 15 October 1986 (CASASENT, Rozzi).

38. "Optical Pattern Recognition and Artificial Intelligence", Proc. 5th Intl. Congress on Applications of Lasers and Electro-Optics, ICALEO'86, Laser Institute of America, pp. 175-184, 10-13 November 1986, Arlington, VA (CASASENT).

39. "Optical Data Processing", Article in *Accent on Research* (Carnegie Mellon University Magazine), p. 23, 1986 (CASASENT).

40. "Spatial-Temporal Correlation Filter for In-Plane Distortion Invariance", Applied Optics, Vol. 25, pp. 4466-4472, 1 December 1986 (Mahalanobis, Kumar, CASASENT).

1987

41. "Knowledge in Optical Symbolic Pattern Recognition Processors", Optical Engineering, *Special Issue on Optical Computing and Nonlinear Optical Signal Processing*, Vol. 26, pp. 34-40, January 1987 (CASASENT, Botha).

42. "A Directed Graph Optical Processor", Proc. SPIE, Vol. 752, pp. 58-71, January 1987 (Baranoski, CASASENT).

43. "Parameter Selection for Iconic and Symbolic Pattern Recognition Filters", Proc. SPIE, Vol. 754, pp. 284-303, January 1987 (CASASENT, Mahalanobis, Liebowitz).

44. "Optical Pattern Recognition and Artificial Intelligence: A Review", Proc. SPIE, Vol. 754, pp. 2-11, January 1987 (CASASENT).

45. "Optical Pattern Recognition and AI Algorithms and Architectures for ATR and Computer Vision", Proc. SPIE, Vol. 755, pp. 83-93, January 1987 (CASASENT).

46. "Electro Optic Target Detection and Object Recognition", Proc. SPIE, Vol. 762, pp. 104-125, January 1987 (CASASENT).

47. "Multiple Degree of Freedom Optical Pattern Recognition", Jet Propulsion Laboratory/NASA Workshop on Space Telerobotics (January 1987), Pasadena, California, JPL Publication 87-13, Volume II, pp. 65-70, ed. G. Rodriguez, pub. 1 July 1987 (CASASENT).

48. "Computer Generated Hologram Recording Using a Laser Printer", Applied Optics, Vol. 26, pp. 136-138, 1 January 1987 (Lee, CASASENT).

49. "Optical Information Processing", Sixth Edition, *Encyclopedia of Science and Technology*, McGraw-Hill, Accepted (CASASENT).

50. "Curved Object Location by Hough Transformations and Inversions", Pattern Recognition, Vol. 20, No. 2, 1987, pp. 181-188 (CASASENT, Krishnapuram).

51. "Real-Time Deformation Invariant Optical Pattern Recognition Using Coordinate Transformations", Applied Optics, Vol 26, pp. 938-942, 15 March 1987 (CASASENT, Xia, Lee, Song).

52. "Error Correction Coding in an Associative Processor", Applied Optics, Vol. 26, pp. 999-1006, 15 March 1987 (Liebowitz, CASASENT).

53. "Rule-Based, Probabilistic, Symbolic Target Classification by Object Segmentation", *Topical Meeting on Optical Computing* (March 1987), Technical Digest Series 1987, Vol. 11 (Optical Society of America, Washington, D.C. 1987), pp. 155-158 (CASASENT, Mahalanobis).

54. "Model-Based Knowledge-Based Optical Processors", Applied Optics, Vol. 26, pp. 1935-1942, 15 May 1987 (CASASENT, Liebowitz).

55. "Hough Space Transformations for Discrimination and Distortion Estimation", Computer Vision, Graphics, and Image Processing, Vol. 38, pp. 299-316, June 1987 (Krishnapuram, CASASENT).

56. "Optical Processing Techniques for Inspection and Automation", Proc. MVA VISION'87, *Society of Manufacturing Engineers (SME)*, pp. 7.1 - 7.8, June 1987, Detroit (CASASENT).

57. "Optical Iconic Filters for Large Class Recognition", Applied Optics, Vol. 26, pp. 2266-2273, 1 June 1987 (CASASENT, Mahalanobis).

58. "Optical Associative Processor for General Linear Transformations", Applied Optics, Vol. 26, pp. 3641-3648, 1 September 1987 (Krishnapuram, CASASENT).

59. "Minimum Average Correlation Energy (MACE) Filters", Applied Optics, Vol. 26, pp. 3633-3640, 1 September 1987 (Mahalanobis, Kumar, CASASENT).

60. "Multi-Functional Optical Logic, Numerical and Pattern Recognition Processor", Proc. AIAA Computers in Aerospace VI Conference, October 1987, pp. 213-218, (CASASENT, Botha).

61. "Rule-Based String Code Processor", Proc. SPIE, Vol. 848, pp. 2-12, November 1987 (CASASENT, Chien).

62. "Optical Feature Extraction for High-Speed Inspection", Proc. SPIE, Vol. 848, pp. 13-24, November 1987 (Clark, CASASENT).

63. "Associative Memory Synthesis, Performance, Storage Capacity and Updating: New Heteroassociative Memory Results", Proc. SPIE, Vol. 848, pp. 313-333, November 1987 (CASASENT, Telfer).

64. "Multi-Sensor Processing: Object Detection and Identification", Proc. SPIE, Vol. 852, pp. 54-71, November 1987 (CASASENT, Liebowitz).

65. "Rule-Based Symbolic Processor for Object Recognition", Applied Optics, Vol. 26, pp. 4795-4802, 15 November 1987 (CASASENT, Mahalanobis).

66. *Optics in Education - A Guide to Optics Programs in North America*, SPIE, 1987-88 Edition.

67. "Adaptive Learning Optical Symbolic Processor", Proc. SPIE, Vol. 882, pp. 30-42, January 1988 (CASASENT, Mahalanobis).

68. "Optical Associative Processors for Visual Perception", Proc. SPIE, Vol. 882, pp. 47-59, January 1988 (CASASENT, Telfer).

69. "Experimental Comparison of Computer Generated Holograms", Proc. SPIE, Vol. 884, pp. 72-80, January 1988 (Han, CASASENT).

70. "Directed Graph for Adaptive Organization and Learning of a Knowledge Base", Applied Optics, Vol. 27, pp. 534-540, 15 February 1988 (CASASENT, Baranoski).

71. "Optical Laboratory Comparison of Computer Generated Holograms for Correlation Matched Spatial Filters", Proc. SPIE, Vol. 938, pp. 15-28, April 1988 (CASASENT, Han).

72. "Optical Laboratory Realization of Distortion Invariant Filters", Proc. SPIE, Vol. 939, pp. 105-120, April 1988 (CASASENT, Ye).

73. "Multiple Degree of Freedom Object Recognition Using Optical Relational Graph Decision Nets", Applied Optics, Vol. 27, pp. 1886-1893, 1 May 1988 (CASASENT, Lee).

74. "Hough Transform Projections and Slices for Object Discrimination and Distortion Estimation", Applied Optics, Vol. 27, pp. 3451-3460, 15 August 1988 (Krishnapuram, CASASENT).

75. "Rule-Based Processing for String Code Identification", Proc. SPIE, Vol. 974, pp. 224-236, August 1988, San Diego (CASASENT, Chien).

76. "Advanced Optical Processors for Multiple Degree-of-Freedom Object Recognition", IEEE Trans. Aerospace and Electronic Systems, Vol. 24, No. 5, pp. 608-617, September 1988 (CASASENT).

77. "Ho-Kashyap Associative Processors", Proc. SPIE, Vol. 1005, November 1988 (Telfer, CASASENT).

78. "Updating Optical Pseudoinverse Associative Memories", Applied Optics, submitted June 1988 (Telfer, CASASENT).

79. "Correlation Filters for Target Detection in a Markov Model Background Clutter", Applied Optics, submitted July 1988 (Kumar, CASASENT, Mahalanobis).

80. "Key and Recollection Vector Effects on Heteroassociative Memory Performance", Applied Optics, Vol. 28, pp. 272-283, 15 January 1989) (CASASENT, Telfer).

81. "Optical Linear Discriminant Functions", Optics Communications, submitted January 1989 (CASASENT, Song).

82. "A Closure Associative Processor", submitted, *IEEE International Conference on Neural Networks*, June 1989, Washington, D.C. (Telfer, CASASENT).

11.2 AFOSR RESEARCH PRESENTATIONS (August 1984-April 1989)

August 1984

1. SPIE Conference - San Diego, California, "Hierarchical Fisher and Moment-Based Pattern Recognition".

2. SPIE Conference - San Diego, California, "SDF Control of Correlation Plane Structure for 3-D Object Representation and Recognition".

3. ASME Conference - Nevada, "Hierarchical Pattern Recognition Using Moment Features".

4. Naval Weapons Center - China Lake, California, "Optical Pattern Recognition for Autonomous Terminal Homing".

September 1984

5. Philips Research Laboratories - Briarcliff, NY - "Optics and Pattern Recognition in Robotics".

6. Optical Society of America - Pittsburgh, PA, "CMU Center for Excellence in Optical Data Processing".

7. Carnegie-Mellon University, ECE Graduate Seminar - Pittsburgh, PA, "Optical Processing Research in the Center for Excellence in Optical Data Processing".

8. Westinghouse Corporation - Baltimore, MD, "Research and Facilities in the Center for Excellence in Optical Data Processing".

October 1984

9. Washington, D.C., "Optical Pattern Recognition: Feature Extraction".

10. Washington, D.C., "Optical Pattern Recognition: Correlators".

11. Washington, D.C., "Synthetic Discriminant Function Case Studies".

12. Carnegie-Mellon University, ECE Sophomore Seminar - Pittsburgh, Pennsylvania, "Research in the Center for Excellence in Optical Data Processing".

13. University of Pittsburgh, Center for Multivariate Analysis - Pittsburgh, PA, "Advanced Multi-Class Distortion-Invariant Pattern Recognition".

14. Wright Patterson Air Force Base - Ohio, "Multi-Functional Optical Signal Processor for Electronic Warfare".

15. George Mason University - Washington, D.C., "Optical Information Processing".

16. SPIE (IOCC) Conference - Boston, MA, "Optimal Linear Discriminant Functions".

November 1984

17. SPIE Robotics Conference - Boston, MA, "Chord Distributions in Pattern Recognition".

January 1985

18. Fairchild Weston - Long Island, NY, "Optical Pattern Recognition and Optical Processing".

19. SPIE Conference - Los Angeles, CA, "Hybrid Optical/Digital Image Pattern Recognition: A Review".

20. SPIE Conference - Los Angeles, CA, "A Computer Generated Hologram for Diffraction-Pattern Sampling".

21. SPIE Conference - Los Angeles, CA, "A Recent Review of Holography in Coherent Optical Pattern Recognition".

22. Sandia National Laboratories - Albuquerque, NM, "Optical Pattern Recognition and Optical Processing".

March 1985

23. Lockheed Missiles & Space Co. - Sunnyvale, CA, "Advanced Hybrid Optical/Digital Pattern Recognition"

24. OSA Topical Meeting on Optical Computing - Lake Tahoe, NV, "Fabrication and Testing of a Space and Frequency-Multiplexed Optical Linear Algebra Processor".

25. OSA Topical Meeting on Machine Vision - Lake Tahoe, NV, "Hierarchical Feature-Based Object Identification".

26. OSA Topical Meeting on Machine Vision - Lake Tahoe, NV, "Correlation Filters for Distortion-Invariance and Discrimination".

27. Texas Instruments - Dallas, TX, "Optical Pattern Recognition".

April 1985

28. Electro-Com Automation, Inc. - Dallas, TX, "Optical Pattern Recognition".

May 1985

29. Carnegie-Mellon University - Board of Trustees, "Optical Data Processing".

August 1985

30. Global Holonetics Corporation - Fairfield, IA "Optical Pattern Recognition".

31. SPIE - San Diego, CA, "Correlation Synthetic Discriminant Functions for Object Recognition and Classification in High Clutter".

September 1985

32. Carnegie-Mellon University, ECE Graduate Seminar - Pittsburgh, PA, "Optical Data Processing at CMU".

33. General Electric Group, Presented at CMU - Pittsburgh, PA, "Optical Data Processing at CMU".

34. SPIE - Cambridge, MA, "Parameter Estimation and In-Plane Distortion Invariant Chord Processing".

35. SPIE - Cambridge, MA, "Optical Processing Techniques for Advanced Intelligent Robots and Computer Vision".

36. SPIE - Cambridge, MA, "High-Dimensionality Feature-Space Processing with Computer Generated Holograms".

October 1985

37. Martin Marietta - Denver, CO, "Optical Data Processing".

November 1985

38. IEEE Computer Society, *Workshop on Computer Architectures for Pattern Analysis and Image Database Management* - Miami Beach, FL, "Optical Computer Architectures for Pattern Analysis".

January 1986

39. SPIE - Los Angeles, CA, "A Feature Space Rule-Based Optical Relational Graph Processor".

40. SPIE - Los Angeles, CA, "Optical Linear Algebra Processors: Architectures and Algorithms".

41. SPIE - Los Angeles, CA, "Optical AI Symbolic Correlators: Architecture and Filter Considerations".

42. Optical Society of America - Los Angeles, CA, "Optical Computing".

43. Corporate Advisory Group on Optical Information Processing - Los Angeles, CA, "Optical Computing".

44. Jet Propulsion Laboratory/NASA - Pasadena, CA, "Optical Linear Algebra and Pattern Recognition Processors".

February 1986

45. Computer Science Department, Carnegie-Mellon University - Pittsburgh, PA, "Optical AI Pattern Recognition Research in ECE".

March 1986

46. Carnegie-Mellon University, Professional Education Program - Pittsburgh, Pennsylvania, "Optical Data Processing".

47. Air Force Institute of Technology - Dayton, Ohio, "Optical Data Processing at Carnegie-Mellon University".

48. Mars Electronics - Philadelphia, PA, "Optical Pattern Recognition".

49. SPIE Advanced Institute Series on Hybrid and Optical Computers - Leesburg, Virginia, "Scene Analysis Research: Optical Pattern Recognition and Artificial Intelligence".

April 1986

50. SPIE - Orlando, FL, "Model-Based System for On-Line Affine Image Transformations".

51. Robotics Institute - Carnegie-Mellon University - Pittsburgh, PA, "Optical AI Pattern Recognition Research in ECE".

May 1986

52. IBM, Federal Systems Division - Manassas, VA, "Optical Computing".

53. Litton Data Systems - Van Nuys, CA, "Multiple Degree of Freedom Pattern Recognition".

54. NASA Jet Propulsion Laboratory, California Institute of Technology - Pasadena, CA, "Multiple Degree of Freedom Optical Pattern Recognition".

July 1986

55. IOCC Conference - Jerusalem, Israel, "Optical Artificial Intelligence Processors".

August 1986

56. SPIE Conference - San Diego, CA, "Distortion-Invariant Associative Processors".

September 1986

57. ALCOA - Pittsburgh, PA, "Optical Information Processing".

58. Eikonix Corp. - Boston, MA, "Optical Pattern Recognition for Optical Character Recognition".

59. Penn State University - State College, PA, "Optical Scene Analysis and Artificial Intelligence".

October 1986

60. SPIE Conference - Boston, MA, "Hierarchical Processor and Matched Filters for Range Image Processing".

61. SPIE Conference - Boston, MA, "Large Class Iconic Pattern Recognition: An OCR Case Study".

62. Carnegie Mellon University, ECE Graduate Seminar - Pittsburgh, PA, "Optical Computing in ECE: 1986".

November 1986

63. ICALEO'86 - Arlington, VA, "Advanced Optical Pattern Recognition and Artificial Intelligence".

64. Optical Society of America (San Diego Chapter) - San Diego, CA, "Optical Computing".

January 1987

65. SPIE Conference - Los Angeles, CA, "A Directed Graph Optical Processor".

66. SPIE Conference - Los Angeles, CA, "Parameter Selection for Iconic and Symbolic Pattern Recognition Filters".

67. SPIE Conference - Los Angeles, CA, "Optical Pattern Recognition and Artificial Intelligence: A Review" (Invited/Keynote Speaker).

68. SPIE Conference - Los Angeles, CA, "Optical Pattern Recognition and AI Algorithms and Architectures for ATR and Computer Vision" (Invited).

69. SPIE Conference - Los Angeles, CA, "Electro Optic Target Detection and Object Recognition" (Invited).

70. Workshop on Space Telerobotics - NASA/JPL, Pasadena, CA, "Multiple Degree of Freedom Optical Pattern Recognition".

71. Hewlett Packard - Palo Alto, CA, "Optical Computing".

February 1987

72. ISC Defense Systems, Inc. - Lancaster, PA, "Optical Computing and Signal Processing".

March 1987

73. OSA Topical Meeting on Optical Computing - Lake Tahoe, NV, "Rule-Based, Probabilistic, Symbolic Target Classification by Object Segmentation".

May 1987

74. NASA Langley Research Center - Hampton, VA, "Overview of Photonics Technology".

75. NASA Langley Research Center - Hampton, VA, "Machine Vision".

June 1987

76. MVA VISION'87 Conference - Detroit, MI, "Optical Processing Techniques for Inspection and Automation".

77. Perkin-Elmer - White Plains, NY, "Optical Computing".

July 1987

78. Galileo - Sturbridge, MA, "Optical Computing".

79. Carnegie Mellon University - ECE Department, Presentation to the attendees of the Fault Tolerant Computing Conference, Pittsburgh, PA.

August 1987

80. UCLA Extension Course - Los Angeles, CA, "Optical Computing".

81. Galileo - Sturbridge, MA, "Product Opportunities in Optical Data Processing".

September 1987

82. Defense Science Board, Pentagon - Washington, D.C, "Optical Computing for Automatic Target Recognition".

November 1987

83. SPIE Robotics Conference - Boston, MA, "Associative Memory Synthesis, Performance, Storage Capacity and Updating: New Heteroassociative Memory Results".

84. SPIE Robotics Conference - Boston, MA, "Rule-Based String Code Processor".

85. SPIE Robotics Conference - Boston, MA, "Multi-Sensor Processing: Object Detection and Identification".

December 1987

86. National Security Agency - Maryland, "Optical Information Processing".

January 1988

87. SPIE - Los Angeles, CA, "Adaptive Learning Optical Symbolic Processor".

88. SPIE - Los Angeles, CA, "Optical Associative Processors for Visual Perception".

89. SPIE - Los Angeles, CA, "Optical Linear Heterodyne Matrix-Vector Processor".

90. SPIE - Los Angeles, CA, "Experimental Comparison of Computer Generated Holograms".

91. Ovionic Imaging Systems - Detroit, MI, "Optical Computing"

92. Hughes - CA, Optical Information Processing.

93. Ford Motor Company - Palo ALto, CA, "Optical Processing"

March 1988

94. E.I. duPont de Nemours and Co. - Wilmington, DE, "Optical Information Processing".

95. Carnegie-Mellon University, Professional Education Program - Pittsburgh, Pennsylvania, "Optical Data Processing".

April 1988

96. SUNY at Binghamton, Photonics Symposium - Binghamton, NY, "Optical Computing: A Review".

97. SPIE - Orlando, FL, "Optical Laboratory Comparison of Computer Generated Holograms for Correlation Matched Spatial Filters".

May 1988

98. Ford Aerospace - Palo Alto, CA, "Optical Information Processing".

June 1988

99. Air Force Office of Scientific Research - Washington, D.C., "Optical Directed Graphs and Associative Processors".

July 1988

100. San Diego International Conference on Neural Networks - San Diego, CA, "Review of Pattern Recognition" (invited tutorial)

August 1988

101. SPIE Conference - San Diego, CA. "Rule-Based Processing for String Code Identification".

November 1988

102. SPIE Conference - Boston, MA, "Ho-Kashyap Associative Processors".

103. Amoco Corporation - CMU, "Optical Computing".

December 1988

104. Itran Corporation - CMU, "Optical Computing".

105. Amoco Corporation - Naperville, IL, "Optical Computing".

11.3 STUDENTS SUPPORTED AND DEGREES AWARDED UNDER AFOSR SUPPORT

1. Eugene Pochapsky, M.S. Dissertation, "The Simulation of Optical Pattern Recognition Systems", September 1984.

2. William Rozzi, M.S. Dissertation, "Advanced Quantitative Synthetic Discriminant Function Tests on Ship Imagery", December 1984.

3. James Fisher, M.S. Dissertation, "Extended Kalman Filter Algorithms for Implementation on a High-Accuracy Optical Processor", December 1984.

4. W.T. Chang, Ph.D. Dissertation, "Chord Distributions and Correlation SDFs in Pattern Recognition", March 1985.

5. Abhijit Mahalanobis, M.S. Dissertation, "Application of Synthetic Discriminant Functions for Optical Character Recognition", September 1985.

6. Suzanne A. Liebowitz, M.S. Dissertation, "Techniques for Multiple-Degree-of-Freedom Processing Using Range Imagery", October 1985.

7. Andrew J. Lee, M S. Dissertation, "High-Dimensionality Feature Space Pattern Recognition Using Computer Generated Holograms", January 1986.

8. Jeffrey Richards, M.S. Dissertation, "Optical Processing for Product Inspection", November 1986.

9. Brian Telfer, M.S. Dissertation, "Optical Associative Memories for Distortion-Invariant Pattern Recognition", February 1987.

10. Chien-Wei Han, M.S. Dissertation, "Experimental Analysis of Error Effects on Computer Generated Hologram Performance", June 1987.

11. Raghuram Krishnapuram, Ph.D. Dissertation, "Hough-Space Associative-Processor for Pattern Recognition", August 1987.

12. Abhijit Mahalanobis, Ph.D. Dissertation, "New Correlation Filters for Symbolic Rule-Based Pattern Recognition", August 1987.

13. Sung-Il Chien, Ph.D. Dissertation "Optical String Code Rule Based and Associative Processors", August 1988.